IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

FINJAN SOFTWARE, LTD., an Israel corporation, Plaintiff-Counterdefendant, v. SECURE COMPUTING CORPORATION, a Delaware corporation, CYBERGUARD, CORPORATION, a Delaware corporation, WEBWASHER AG, a German corporation and DOES 1 THROUGH 100, Defendants-Counterclaimants.	)	C. A. No. 06-369 (GMS) **PUBLIC VERSION**

JOINT APPENDIX OF INTRINSIC AND EXTRINSIC EVIDENCE RE: CLAIM CONSTRUCTION BRIEFING

VOLUME 1

Philip A. Rovner (#3215)
POTTER ANDERSON & CORROON LLP
Hercules Plaza
P. O. Box 951
Wilmington, DE 19899
(303) 984-6000
provner@potteranderson.com

Attorneys for Plaintiff
Finjan Software, Ltd.

OF COUNSEL:

Paul J. Andre
Lisa Kobialka
Meghan A. Wharton
James R. Hannah
Perkins Coie LLP
101 Jefferson Drive
Menlo Park, CA 94025-1114
(650) 838-4300

Dated: September 28, 2007
Public Version: October 4, 2007

JOINT APPENDIX OF INTRINSIC AND EXTRINSIC EVIDENCE

Tab	Description	Party Citing	Page(s)
1	U.S. Patent No. 6,092,194	Finjan Secure Computing	JA1 – JA20
2	U.S. Patent No. 6,804,780	Finjan Secure Computing	JA21 - JA38
3	U.S. Patent No. 7,058,822	Finjan Secure Computing	JA39 - JA62
4	U.S. Patent No. 6,357,010	Finjan Secure Computing	JA63 – JA83
5	U.S. Patent No. 7,185,361	Finjan Secure Computing	JA84 - JA93
6	'194 Patent, Office Action dated January 7, 1999	Finjan	JA181 - JA190
7	'194 Patent, Office Action mailed June 7, 1999	Finjan	JA207 – JA215
8	'194 Patent, Response to Office Action, dated October 27, 1999	Finjan	JA228 - JA234
9	'194 Patent, Information Disclosure Citation	Finjan	JA250
10	'780 Patent, Response to Office Action, dated July 31, 2003	Finjan	JA404 – JA412
11	'010 Patent, Response to Office Action, dated September 19, 2000	Finjan	JA926 - JA947

12	'361 Patent, Office Action mailed September 10, 2003	Finjan	JA1087 – JA1098
13	'361 Patent, Response to Office Action, dated January 12, 2004	Finjan	JA1100 – JA1111
14	'361 Patent, Final Rejection/Office Action, mailed on February 18, 2004	Finjan	JA1113 – JA1124
15	'361 Patent, Appellants' Brief on Appeal Under 37 C.F.R.41.37©	Finjan	JA1130 – JA1155
16	Zhang, X.N. *Secure Code Distribution.* June, 1997.	Finjan Secure Computing	JA1228 - JA1232
17	Excerpts from the Deposition of Martin Stecher, Augsut 28 & 29, 2007	Finjan	JA1233 – JA1239
18	Excerpts from the Deposition of Steven O. Chew, September 7, 2007	Finjan	JA1240 – JA1246
19	Excerpts from the Deposition of Frank Berzau, August 31, 2007	Finjan	JA1247 - JA1250
20	Excerpts from the Deposition of Christoph Alme, August 29, 2007	Finjan	JA1251 - JA1258
21	Excerpts from the Deposition of Roland Scholz, August 31, 2007	Finjan	JA1259 – JA1264
22	Excerpts from the Deposition of Michael J. Gallagher, August 3, 2007	Finjan	JA1265 – JA1269
23	Excerpts from the Deposition of Peter Borgolte, August 31, 2007	Finjan	JA1270 – JA1274
24	Excerpts from the Deposition of Jan Schnellbacher, August 31, 2007	Finjan	JA1275 - JA1281
25	U.S. Patent No. 6,167,520	Secure Computing	JA2000 - JA2012
26	Issue Notification to the '361 Patent	Secure Computing	JA2013
27	'361 Patent, Office Action mailed September 10, 2003	Secure Computing	JA2014 - JA2024

28	'361 Patent, January 12, 2004 Response to Office Action	Secure Computing	JA2025 - JA2036
29	'010 Patent, Cover of U.S. Utility Patent Application No. 09/024,576	Secure Computing	JA2037
30	'010 Patent, July 17, 2001 Response to Office Action	Secure Computing	JA2038 – JA2044
31	'194 Patent, Cover of U.S. Utility Patent Application No. 08/964,388	Secure Computing	JA2045
32	'194 Patent, October 27, 1999 Response to Office Action	Secure Computing	JA2046 - JA2052
33	'780 Patent, Cover to U.S. Utility Patent Application No. 09/539,667	Secure Computing	JA2053
34	'780 Patent, July 31, 2003 Response to Office Action	Finjan Secure Computing	JA2054 – JA2062
35	Letter from Forrester & Boehmert to European Patent Office re: European Patent Application No. 97950351.3 (December 21, 2005)	Finjan Secure Computing	JA2063 - JA2067
36	McDaniel, George. *IBM Dictionary of Computing*. (Tenth Edition, 1994)	Secure Computing	JA2068 – JA2072
37	Dowining, Douglas A., Covington, Michael A., & Covington. *Dictionary of Computer and Internet Terms*. (Fifth Edition, 1996)	Secure Computing	JA2073 - JA2076
38	*Dictionary of Computer Words: An A to Z Guide to Today's Computers*. (Revised Edition, 1995)	Secure Computing	JA2077 - JA2079
39	*Epicrealm, Licensing, LLC v. Autoflex Leasing, Inc., et al*. 2006 WL 3099603 (E.D.Tex.)	Secure Computing	JA2080 - JA2095
40	*Collegenet, Inc. v. XAP Corp.* 2004 WL 2429843 (D.Or.)	Secure Computing	JA2096 - JA2132
41	*Pipe Liners, Inc. v. Pipelining Products, Inc.* 1999 WL 1011974 (D.Del.)	Secure Computing	JA2133 – JA2146
42	Excerpts from Deposition transcript of Yuval Ben-Itzhak, August 10, 2007	Secure Computing	JA2147 – JA2152

43	Excerpts from Deposition transcript of Michael J. Gallagher, August 3, 2007	Secure Computing	JA2153 – JA2155
44	Excerpts from Rough Transcript of Deposition of Martin Stecher, August 28, 2007	Secure Computing	JA2156 – JA2158
45	Chappell, David. *Introducing Active X.* (January 1997).	Secure Computing	JA2159 – JA2162
46	Search result from USPTO Trademark Electronic Search System (TESS) on September 7, 2007	Secure Computing	JA2163 – JA2164
47	'361 Patent, Appellants' Brief on Appeal	Secure Computing	JA2165 - JA2190
48	Finjan Software Ltd. – Financial Statements as of December 31, 1998	Secure Computing	JA2191- JA2206
49	Finjan Software Ltd. and Its Subsidiary – Consolidated Financial Statements as of December 31, 1999	Secure Computing	JA2207 - JA2222
50	Finjan Software Ltd.– Consolidated Financial Statements as of December 31, 2000	Secure Computing	JA2223 - JA2243
51	Finjan Software Ltd. and Its Subsidiary – Consolidated Financial Statements as of December 31, 2001	Secure Computing	JA2244 – JA2262
52	Finjan Software, Inc., and Its Subsidiaries – Consolidated Financial Statements as of December 31, 2002	Secure Computing	JA2263 -2281
53	Finjan Software, Inc. and Its Subsidiaries – Consolidated Financial Statements as of December 31, 2003	Secure Computing	JA2282 – 2304
54	Finjan Software, Inc. and Its Subsidiaries – Consolidated Financial Statements as of December 31, 2004	Secure Computing	JA2305 – JA2325
55	Finjan Software Ltd. and Its Subsidiary – Consolidated Financial Statements dated December 31, 2005	Secure Computing	JA2326- JA2347
56	Webster's New World Dictionary of Computer Terms (Fifth Edition)	Secure Computing	JA2348 - JA2350

57	Garner, Bryan A. *Modern American Usage*. Oxford University Press, 2003.	Secure Computing	JA2351-JA2353
58	Secure Computing – Company Fact Sheet	Secure Computing	JA2354 - JA2360
59	Secure Computing – "Sidewinder – The Origin of Sidewinder G2 Security Appliance."	Secure Computing	JA2361 - JA2364
60	U.S. Patent No. 5,864,683	Secure Computing	JA2365 - JA2401
61	Committee Membership Information: Technical Privacy Dimensions of Information for Terrorism Prevention and Other National Goals	Secure Computing	JA2402 - JA2413

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

CERTIFICATE OF SERVICE

I, Philip A. Rovner, hereby certify that on October 4, 2007, the within document was filed with the Clerk of the Court using CM/ECF which will send notification of such filing(s) to the following; that the document was served on the following counsel as indicated; and that the document is available for viewing and downloading from CM/ECF.

BY HAND DELIVERY

Frederick L. Cottrell, III, Esq.
Kelly E. Farnan, Esq.
Richards, Layton & Finger, P.A.
One Rodney Square
920 N. King Street
Wilmington, DE 19801
cottrell@rlf.com; farnan@rlf.com

I hereby certify that on October 4, 2007 I have sent by Federal Express the foregoing document to the following non-registered participants:

Jake M. Holdreith, Esq.
Christopher A. Seidl, Esq.
Robins, Kaplan, Miller & Ciresi L.L.P.
2800 LaSalle Plaza
800 LaSalle Avenue
Minneapolis, MN 55402
jmholdreith@rkmc.com ; caseidl@rkmc.com

/s/ Philip A. Rovner
Philip A. Rovner (#3215)
Potter Anderson & Corroon LLP
Hercules Plaza
P.O. Box 951
Wilmington, Delaware 19899
(302) 984-6000
E-mail: provner@potteranderson.com

US006092194A

United States Patent [19]

Touboul

[11] Patent Number: 6,092,194

[45] Date of Patent: *Jul. 18, 2000

[54] **SYSTEM AND METHOD FOR PROTECTING A COMPUTER AND A NETWORK FROM HOSTILE DOWNLOADABLES**

[75] Inventor: Shlomo Touboul, Kefar-Haim, Israel

[73] Assignee: Finjan Software, Ltd., Netanya, Israel

[*] Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

[21] Appl. No.: 08/964,388

[22] Filed: Nov. 6, 1997

Related U.S. Application Data

[60] Provisional application No. 60/030,639, Nov. 8, 1996.

[51] Int. Cl.[7] H04L 1/00

[52] U.S. Cl. 713/200

[58] Field of Search 395/187.01, 186; 713/200, 201, 202; 714/38, 704; 709/229

[56] References Cited

U.S. PATENT DOCUMENTS

5,077,677	12/1991	Murphy et al.	395/10
5,361,359	11/1994	Tajalli et al.	395/700
5,485,409	1/1996	Gupta et al.	395/186
5,485,575	1/1996	Chess et al.	395/183.14
5,572,643	11/1996	Judson	395/793
5,623,600	4/1997	Ji et al.	395/187.01
5,638,446	6/1997	Rubin	380/25
5,692,047	11/1997	McManis	380/4
5,692,124	11/1997	Holden et al.	395/187.01
5,720,033	2/1998	Deo	395/186
5,724,425	3/1998	Chang et al.	380/25
5,740,248	4/1998	Fieres et al.	380/25
5,761,421	6/1998	van Hoff et al.	395/200.53
5,765,205	6/1998	Breslau et al.	711/203
5,784,459	7/1998	Devarakonda et al.	380/4
5,796,952	8/1998	Davis et al.	395/200.54
5,805,829	9/1998	Cohen et al.	395/200.32
5,832,208	11/1998	Chen et al.	395/187.01
5,850,559	12/1998	Angelo et al.	395/750.03
5,864,683	1/1999	Boebert et al.	395/200.79
5,892,904	4/1999	Atkinson et al.	395/187.01

OTHER PUBLICATIONS

Web page: http://ietlibs.com:80/cgi-bin/ietl_cgi?se...2ehts%26ViewTemplate%3ddocvie%5fb%2ehts, Okamato, E. et al., "ID-Based Authentication System For Computer Virus Detection", IEEE/IEE Electronic Library online, Electronics Letters, vol. 26, Issue 15, ISSN 0013-5194, Jul. 19, 1990, Abstract and pp. 1169-1170.

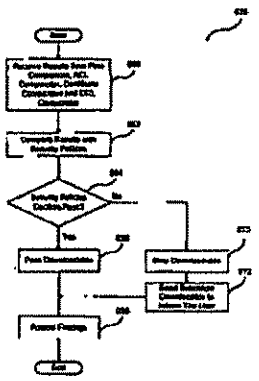
(List continued on next page.)

Primary Examiner—Robert W. Beausoliel, Jr.
Assistant Examiner—Christopher Revak
Attorney, Agent, or Firm—Graham & James LLP

[57] **ABSTRACT**

A system protects a computer from suspicious Downloadables. The system comprises a security policy, an interface for receiving a Downloadable, and a comparator, coupled to the interface, for applying the security policy to the Downloadable to determine if the security policy has been violated. The Downloadable may include a Java™ applet, an ActiveX™ control, a JavaScript™ script, or a Visual Basic script. The security policy may include a default security policy to be applied regardless of the client to whom the Downloadable is addressed, or a specific security policy to be applied based on the client or the group to which the client belongs. The system uses an ID generator to compute a Downloadable ID identifying the Downloadable, preferably, by fetching all components of the Downloadable and performing a hashing function on the Downloadable including the fetched components. Further, the security policy may indicate several tests to perform, including (1) a comparison with known hostile and non-hostile Downloadables; (2) a comparison with Downloadables to be blocked or allowed per administrative override; (3) a comparison of the Downloadable security profile data against access control lists; (4) a comparison of a certificate embodied in the Downloadable against trusted certificates; and (5) a comparison of the URL from which the Downloadable originated against trusted and untrusted URLs. Based on these tests, a logical engine can determine whether to allow or block the Downloadable.

68 Claims, 10 Drawing Sheets



OTHER PUBLICATIONS

"Finjan Announces a Personal Java ™ Firewall For Web Browsers—the SurfinShield™ 1.6", Press Release of Finjan Releases SurfinShield, Oct. 21, 1996, 2 pages.

"Finjan Software Releases SurfinBoard, Industry's First JAVA Security Product For the Wodld Wide Web", Article published on the Internet by Finjan Software, Ltd., Jul. 29, 1996, 1 page.

"Powerful PC Security for the New World of Java™ and Downloadables, Surfin Shield™"Article published on the Internet by Finjan Software Ltd., 1996, 2 Pages.

"Company Profile Finjan—Safe Surfing, The Java Security Solutions Provider" Article published on the Internet by Finjan Software Ltd., Oct. 31, 1996, 3 pages.

"Finjan Announces Major Power Boost and New Features for SurfinShield™ 2.0" Las Vegas Convention Center/Pavillion 5 P5551, Nov. 18, 1996, 3 pages.

"Java Security: Issues & Solutions" Article published on the Internet by Finjan Software Ltd., 1996, 8 pages.

"Products" Article published on the Internet, 7 pages.

Mark LaDue, "Online Business Consultant" Article published on the Internet, Home Page, Inc. 1996, 4 pages.

Jim K. Omura, "Novel Applications of Cryptography in Digital Communications", IEEE Communications Magazine, p 27, May 1990.

Norvin Leach et al, "IE 3.0 applets will earn certification", PC Week, v13, n29, p1(2), Jul. 1996.

Microsoft Authenticode Technology, "Ensuring Accountability and Authenticity for Software Components on the Internet", Microsoft Corporation, Oct. 1996.

Frequently Asked Questions About Authenticode, Microsoft Corporation, Feb. 1997.

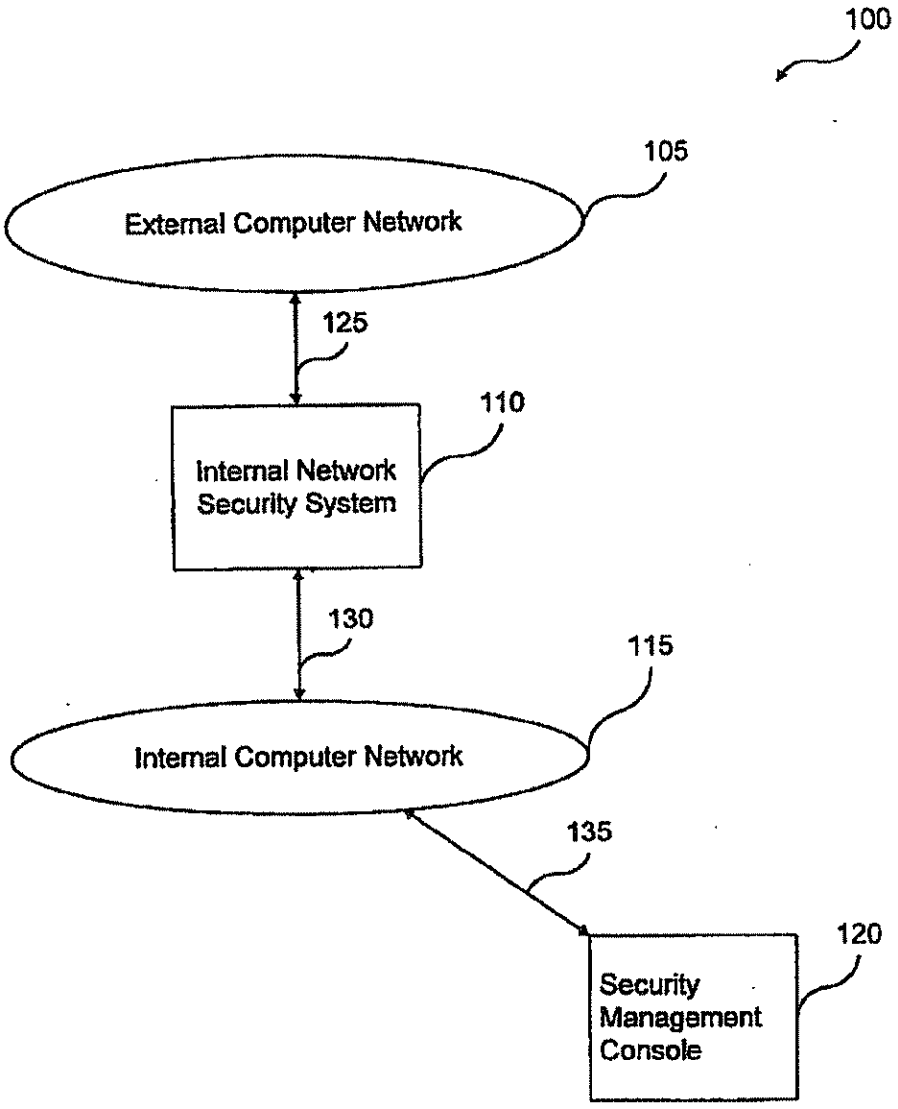

100
105
External Computer Network
125
110
Internal Network Security System
130
115
Internal Computer Network
135
120
Security Management Console

FIG. 1

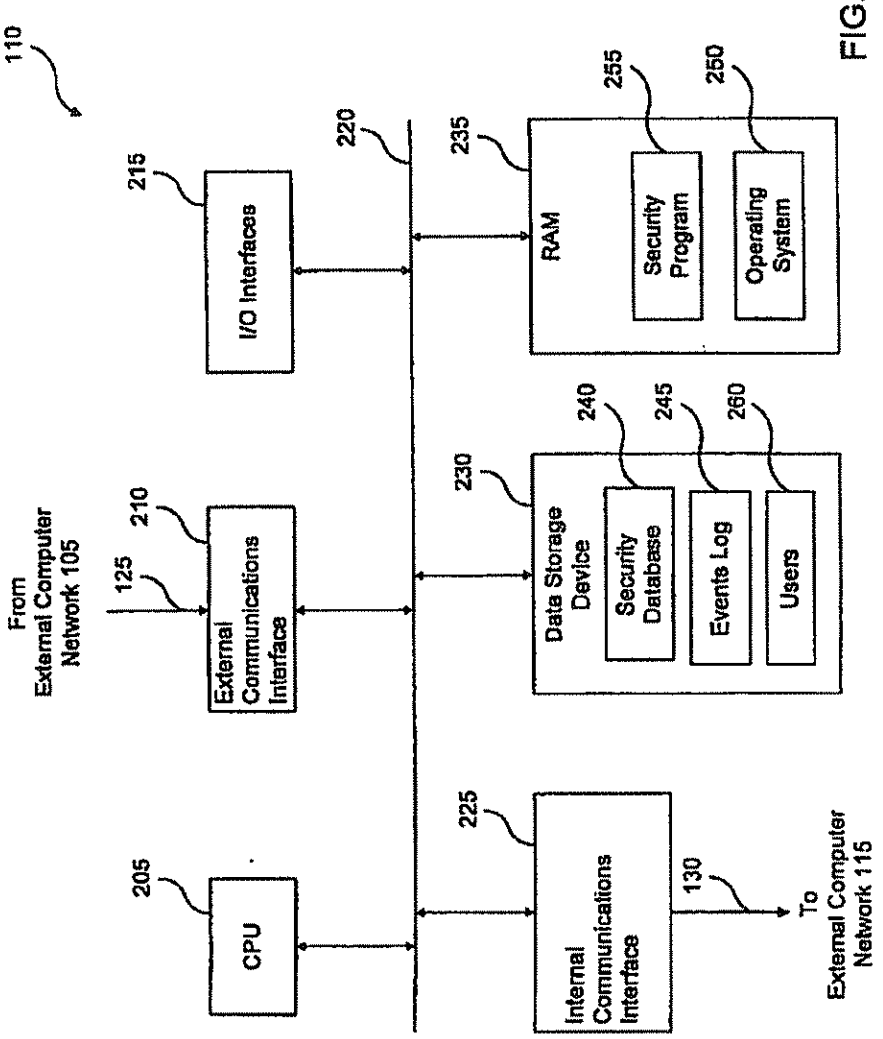

110
215
I/O Interfaces
220
235
RAM
255
Security Program
260
Operating System
From External Computer Network 105
125
210
External Communications Interface
230
Data Storage Device
240
Security Database
245
Events Log
260
Users
205
CPU
225
Internal Communications Interface
130
To External Computer Network 115

FIG. 2

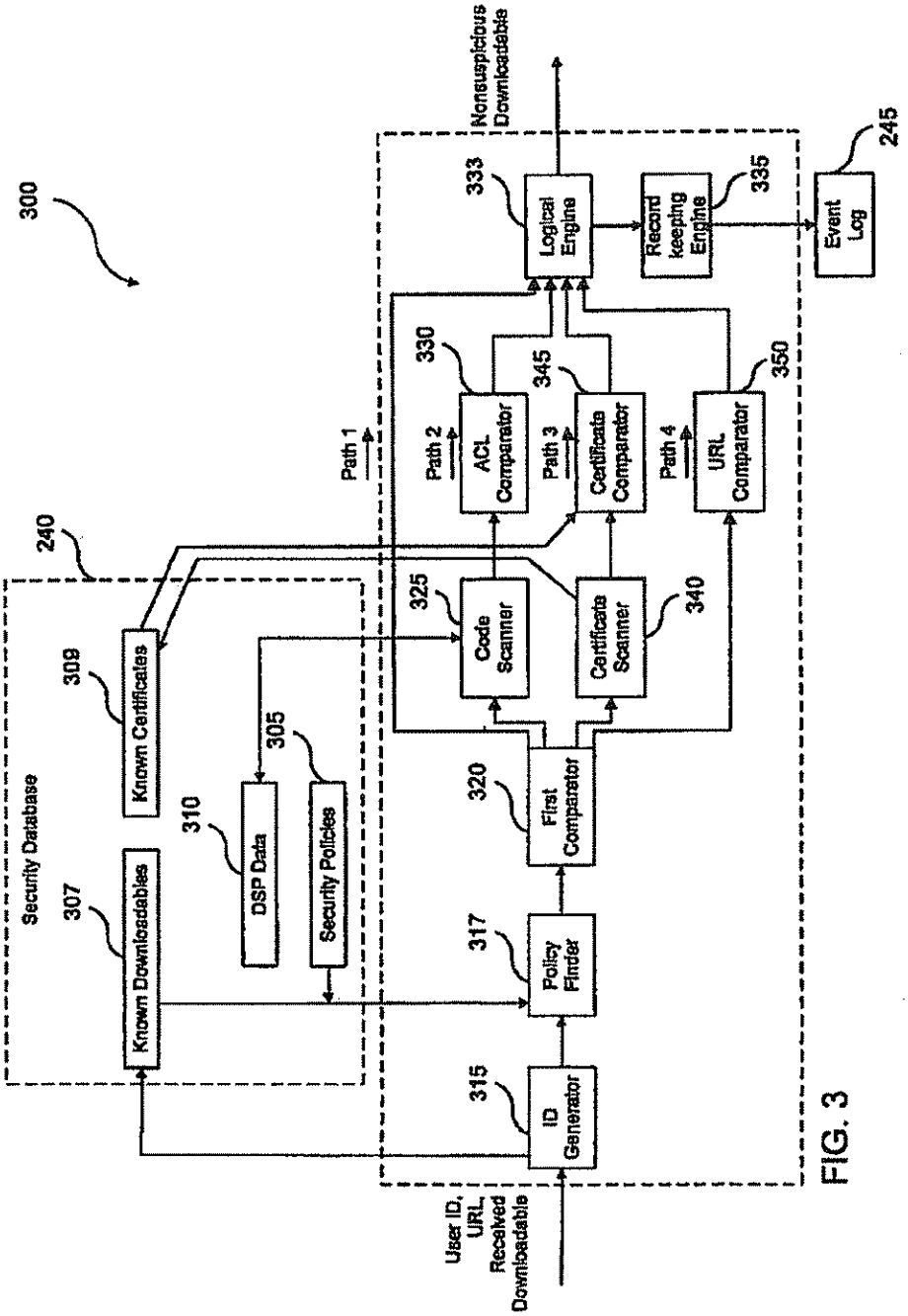

300
Nonsuspicious Downloadable
333
Logical Engine
Record keeping Engine
335
Event Log
245
330
345
350
Path 1
Path 2
ACL Comparator
Path 3
Certificate Comparator
Path 4
URL Comparator
240
325
Code Scanner
Certificate Scanner
340
309
Known Certificates
305
310
DSP Data
Security Policies
320
First Comparator
Security Database
307
Known Downloadables
317
Policy Finder
315
ID Generator
User ID, URL, Received Downloadable

FIG. 3

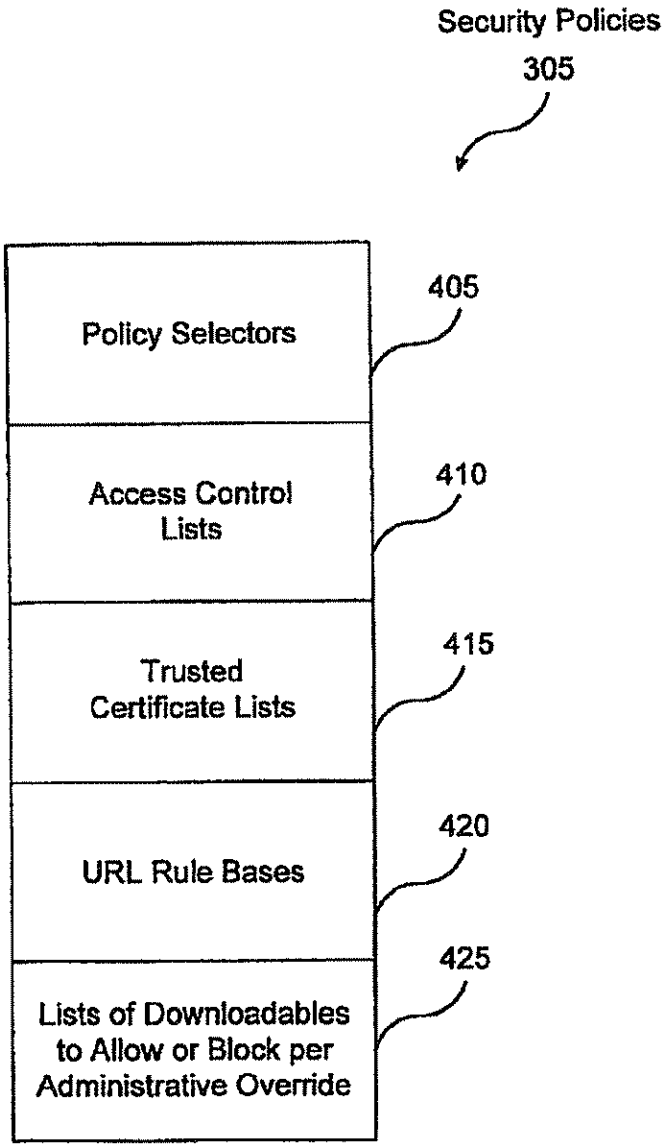

Security Policies
305
Policy Selectors
405
Access Control Lists
410
Trusted Certificate Lists
415
URL Rule Bases
420
Lists of Downloadables to Allow or Block per Administrative Override
425

FIG. 4

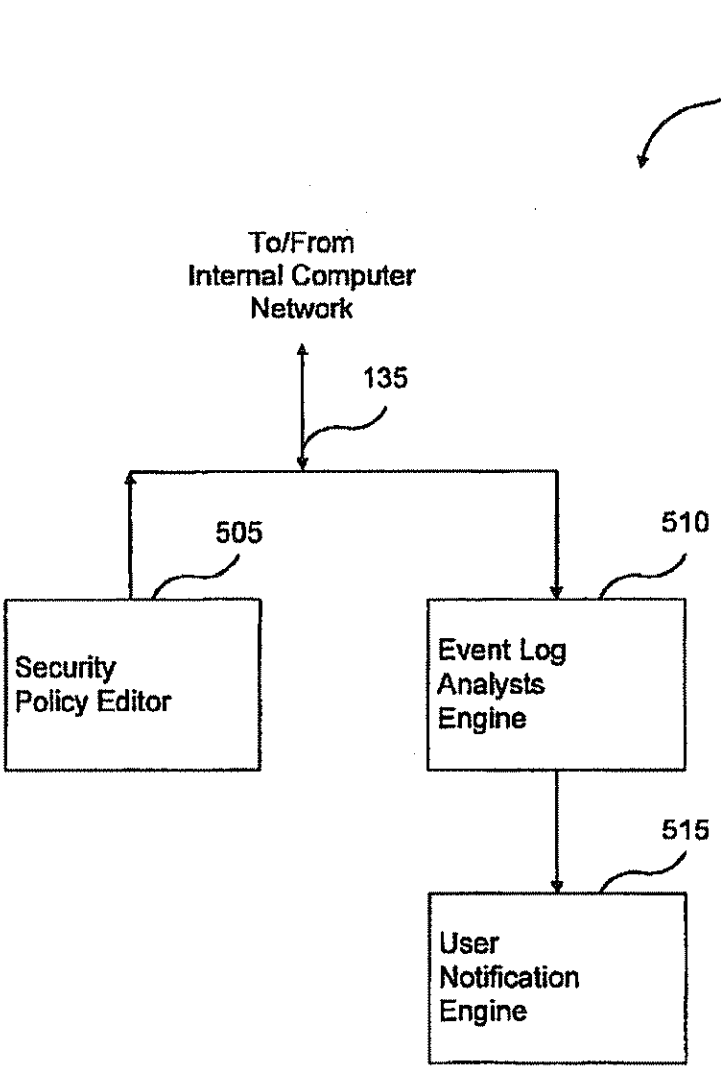

120
To/From
Internal Computer
Network
135
505
510
Security
Policy Editor
Event Log
Analysts
Engine
515
User
Notification
Engine

FIG. 5

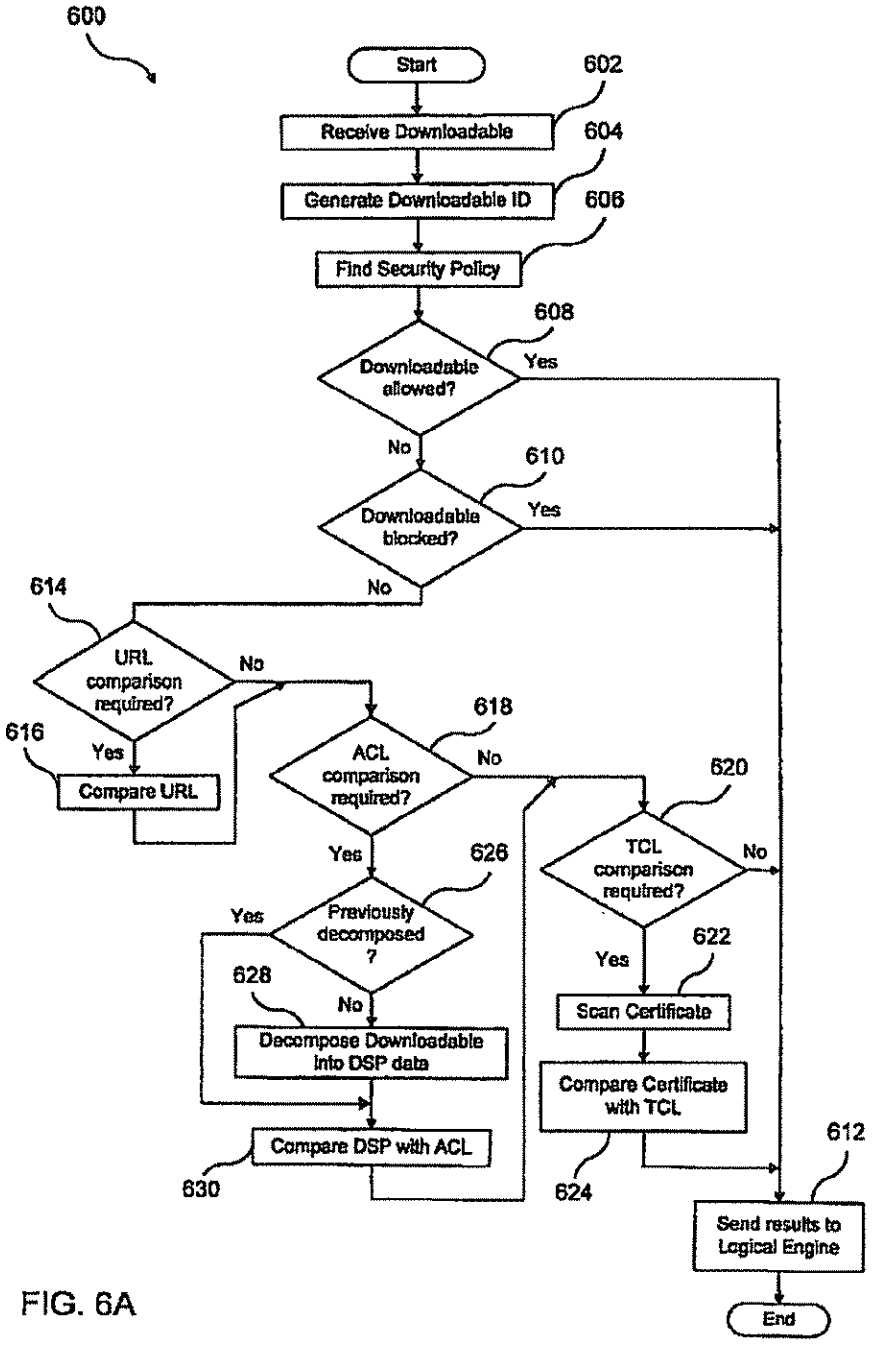

600
Start
602
Receive Downloadable
604
Generate Downloadable ID
606
Find Security Policy
608
Downloadable allowed?
Yes
No
610
Downloadable blocked?
Yes
No
614
URL comparison required?
No
616
Yes
Compare URL
618
ACL comparison required?
No
Yes
626
Previously decomposed ?
Yes
628
No
Decompose Downloadable into DSP data
Compare DSP with ACL
630
620
TCL comparison required?
No
Yes
622
Scan Certificate
Compare Certificate with TCL
624
612
Send results to Logical Engine
End

FIG. 6A

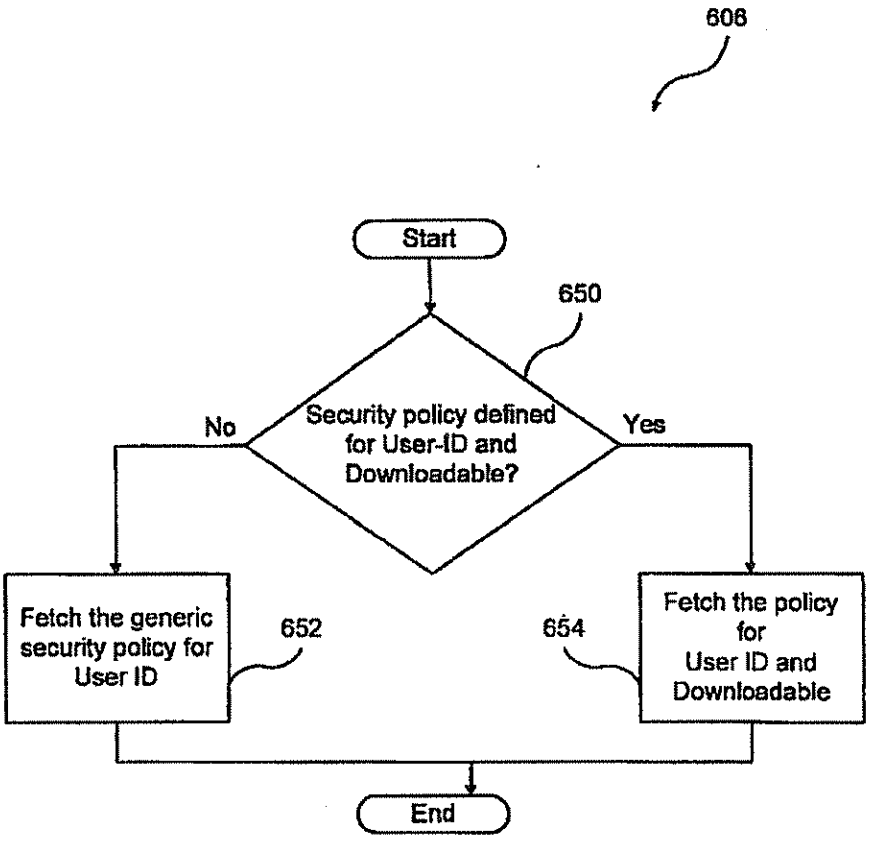

606
Start
650
Security policy defined for User-ID and Downloadable?
No
Yes
Fetch the generic security policy for User ID
652
654
Fetch the policy for User ID and Downloadable
End

FIG. 6B

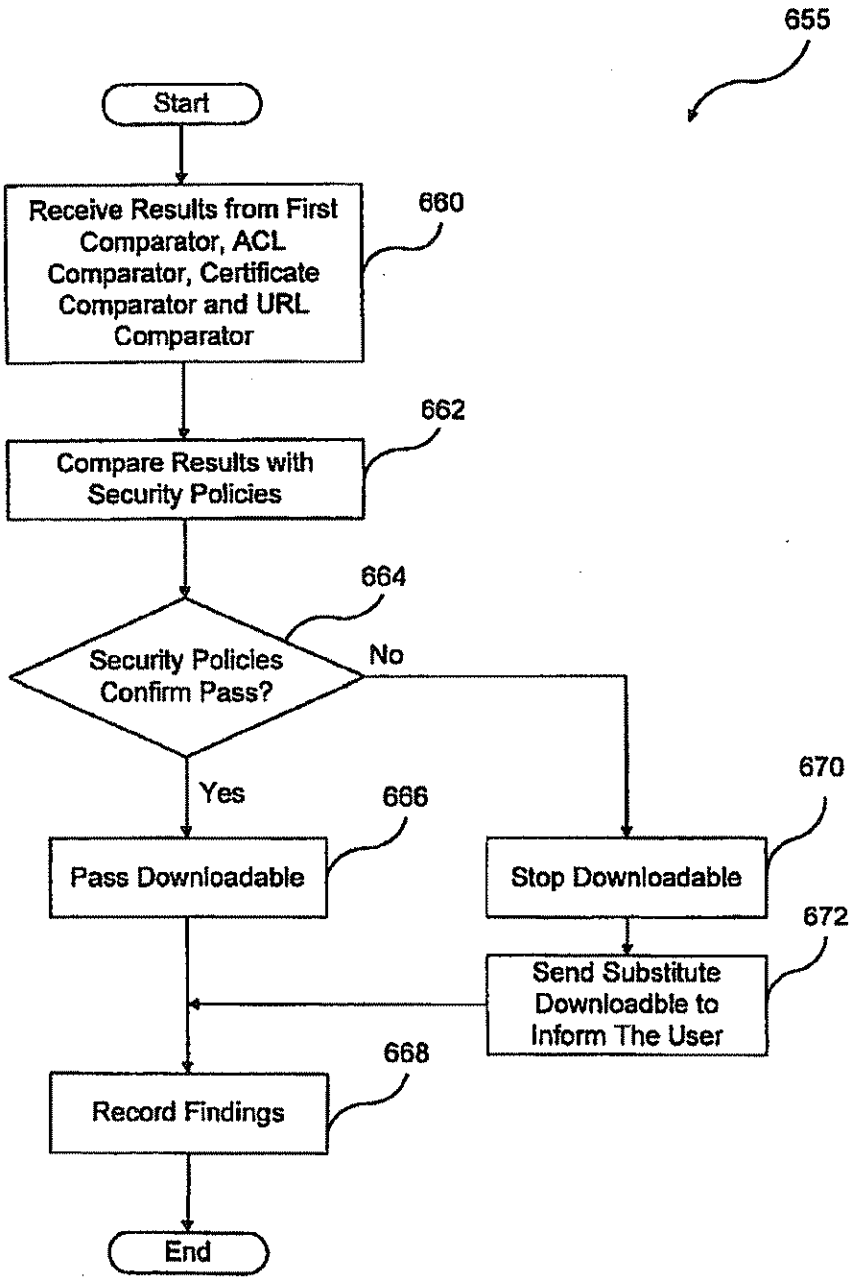

655
Start
Receive Results from First Comparator, ACL Comparator, Certificate Comparator and URL Comparator
660
Compare Results with Security Policies
662
Security Policies Confirm Pass?
664
No
Yes
Pass Downloadable
666
Stop Downloadable
670
Send Substitute Downloadble to Inform The User
672
Record Findings
668
End

FIG. 6C

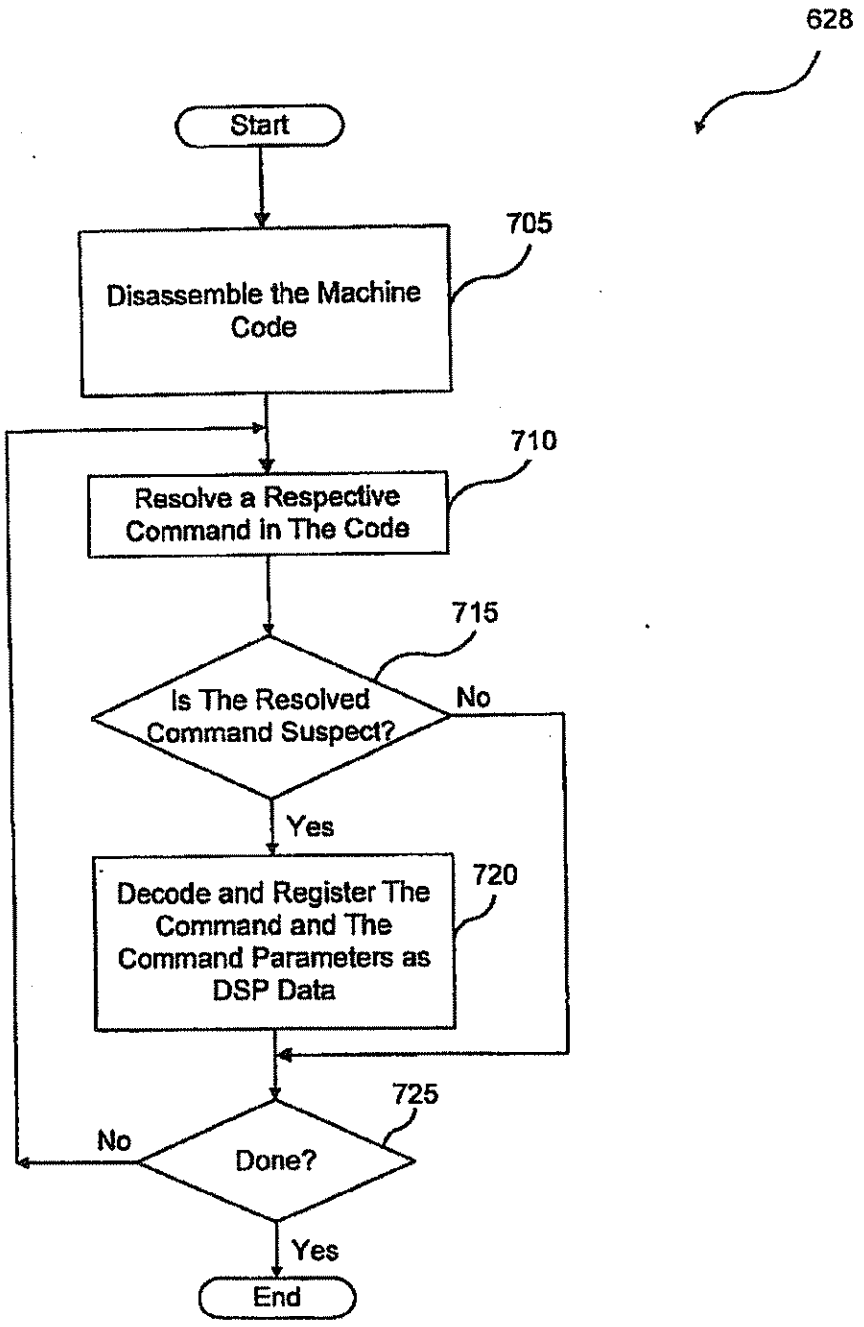

628
Start
705
Disassemble the Machine Code
710
Resolve a Respective Command in The Code
715
Is The Resolved Command Suspect?
No
Yes
720
Decode and Register The Command and The Command Parameters as DSP Data
725
Done?
No
Yes
End

FIG. 7

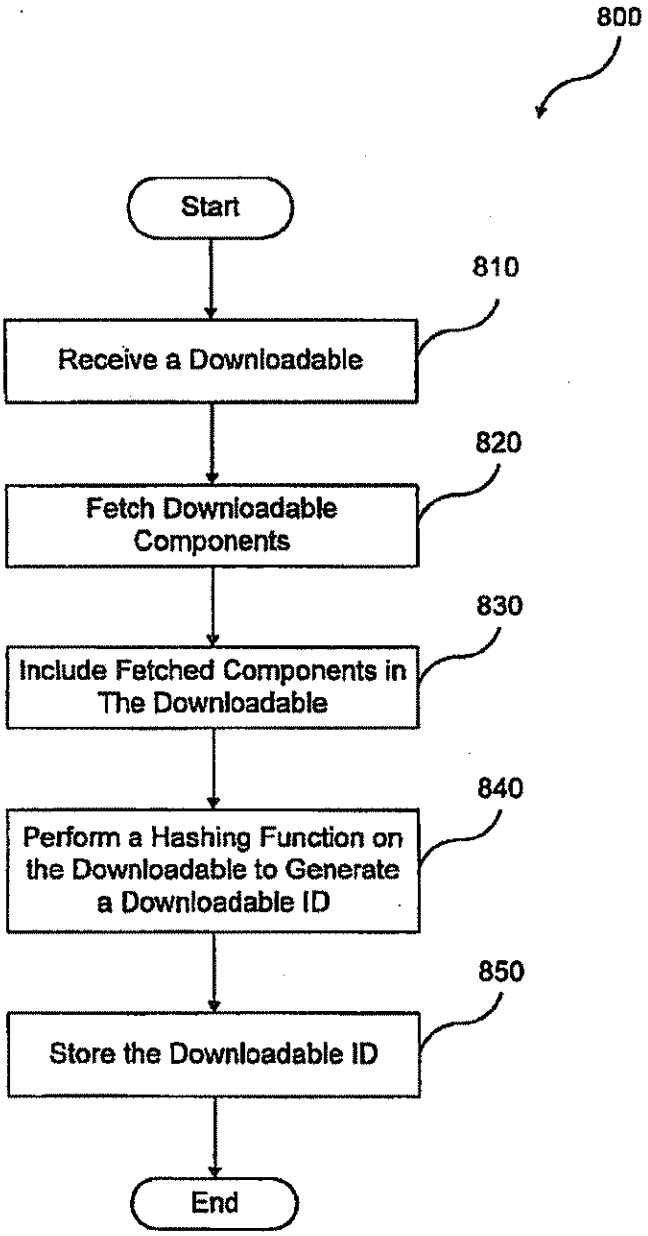

800
Start
810
Receive a Downloadable
820
Fetch Downloadable Components
830
Include Fetched Components in The Downloadable
840
Perform a Hashing Function on the Downloadable to Generate a Downloadable ID
850
Store the Downloadable ID
End

FIG. 8

SYSTEM AND METHOD FOR PROTECTING A COMPUTER AND A NETWORK FROM HOSTILE DOWNLOADABLES

INCORPORATION BY REFERENCE TO RELATED APPLICATION

This application hereby incorporates by reference related U.S. patent application Ser. No. 08/790,097, entitled "System and Method for Protecting a Client from Hostile Downloadables," filed on Jan. 29, 1997, by inventor Shlomo Touboul.

PRIORITY REFERENCE TO PROVISIONAL APPLICATION

This application claims benefit of and hereby incorporates by reference provisional application Ser. No. 60/030,639, entitled "System and Method for Protecting a Computer from Hostile Downloadables," filed on Nov. 8, 1996, by inventor Shlomo Touboul.

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates generally to computer networks, and more particularly provides a system and method for protecting a computer and a network from hostile Downloadables.

2. Description of the Background Art

The Internet is currently a collection of over 100,000 individual computer networks owned by governments, universities, nonprofit groups and companies, and is expanding at an accelerating rate. Because the Internet is public, the Internet has become a major source of many system damaging and system fatal application programs, commonly referred to as "viruses."

Accordingly, programmers continue to design computer and computer network security systems for blocking these viruses from attacking both individual and network computers. On the most part, these security systems have been relatively successful. However, these security systems are not configured to recognize computer viruses which have been attached to or configured as Downloadable application programs, commonly referred to as "Downloadables." A Downloadable is an executable application program, which is downloaded from a source computer and run on the destination computer. Downloadable is typically requested by an ongoing process such as by an Internet browser or web engine. Examples of Downloadables include Java™ applets designed for use in the Java™ distributing environment developed by Sun Microsystems, Inc., JavaScript scripts also developed by Sun Microsystems, Inc., ActiveX™ controls designed for use in the ActiveX™ distributing environment developed by the Microsoft Corporation, and Visual Basic also developed by the Microsoft Corporation. Therefore, a system and method are needed to protect a network from hostile Downloadables.

SUMMARY OF THE INVENTION

The present invention provides a system for protecting a network from suspicious Downloadables. The system comprises a security policy, an interface for receiving a Downloadable, and a comparator, coupled to the interface, for applying the security policy to the Downloadable to determine if the security policy has been violated. The Downloadable may include a Java™ applet, an ActiveX™ control, a JavaScript™ script, or a Visual Basic script. The security policy may include a default security policy to be applied regardless of the client to whom the Downloadable is addressed, a specific security policy to be applied based on the client or the group to which the client belongs, or a specific policy to be applied based on the client/group and on the particular Downloadable received. The system uses an ID generator to compute a Downloadable ID identifying the Downloadable, preferably, by fetching all components of the Downloadable and performing a hashing function on the Downloadable including the fetched components.

Further, the security policy may indicate several tests to perform, including (1) a comparison with known hostile and non-hostile Downloadables; (2) a comparison with Downloadables to be blocked or allowed per administrative override; (3) a comparison of the Downloadable security profile data against access control lists; (4) a comparison of a certificate embodied in the Downloadable against trusted certificates; and (5) a comparison of the URL from which the Downloadable originated against trusted and untrusted URLs. Based on these tests, a logical engine can determine whether to allow or block the Downloadable.

The present invention further provides a method for protecting a computer from suspicious Downloadables. The method comprises the steps of receiving a Downloadable, comparing the Downloadable against a security policy to determine if the security policy has been violated, and discarding the Downloadable if the security policy has been violated.

It will be appreciated that the system and method of the present invention may provide computer protection from known hostile Downloadables. The system and method of the present invention may identify Downloadables that perform operations deemed suspicious. The system and method of the present invention may examine the Downloadable code to determine whether the code contains any suspicious operations, and thus may allow or block the Downloadable accordingly.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram illustrating a network system, in accordance with the present invention;

FIG. 2 is a block diagram illustrating details of the internal network security system of FIG. 1;

FIG. 3 is a block diagram illustrating details of the security program and the security database of FIG. 2;

FIG. 4 is a block diagram illustrating details of the security policies of FIG. 3;

FIG. 5 is a block diagram illustrating details of the security management console of FIG. 1;

FIG. 6A is a flowchart illustrating a method of examining for suspicious Downloadables, in accordance with the present invention;

FIG. 6B is a flowchart illustrating details of the step for finding the appropriate security policy of FIG. 6A;

FIG. 6C is a flowchart illustrating a method for determining whether an incoming Downloadable is to be deemed suspicious;

FIG. 7 is a flowchart illustrating details of the FIG. 6 step of decomposing a Downloadable; and

FIG. 8 is a flowchart illustrating a method 800 for generating a Downloadable ID for identifying a Downloadable.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

FIG. 1 is a block diagram illustrating a network system 100, in accordance with the present invention. The network

system 100 includes an external computer network 105, such as the Wide Area Network (WAN) commonly referred to as the Internet, coupled via a communications channel 125 to an internal network security system 110. The network system 100 further includes an internal computer network 115, such as a corporate Local Area Network (LAN), coupled via a communications channel 130 to the internal network computer system 110 and coupled via a communications channel 135 to a security management console 120.

The internal network security system 110 examines Downloadables received from external computer network 105, and prevents Downloadables deemed suspicious from reaching the internal computer network 115. It will be further appreciated that a Downloadable is deemed suspicious if it performs or may perform any undesirable operation, or if it threatens or may threaten the integrity of an internal computer network 115 component. It is to be understood that the term "suspicious" includes hostile, potentially hostile, undesirable, potentially undesirable, etc. Security management console 120 enables viewing, modification and configuration of the internal network security system 110.

FIG. 2 is a block diagram illustrating details of the internal network security system 110, which includes a Central Processing Unit (CPU) 205, such as an Intel Pentium® microprocessor or a Motorola Power PC® microprocessor, coupled to a signal bus 220. The internal network security system 110 further includes an external communications interface 210 coupled between the communications channel 125 and the signal bus 220 for receiving Downloadables from external computer network 105, and an internal communications interface 225 coupled between the signal bus 220 and the communications channel 130 for forwarding Downloadables not deemed suspicious to the internal computer network 115. The external communications interface 210 and the internal communications interface 225 may be functional components of an integral communications interface (not shown) for both receiving Downloadables from the external computer network 105 and forwarding Downloadables to the internal computer network 115.

Internal network security system 110 further includes Input/Output (I/O) interfaces 215 (such as a keyboard, mouse and Cathode Ray Tube (CRT) display), a data storage device 230 such as a magnetic disk, and a Random-Access Memory (RAM) 235, each coupled to the signal bus 220. The data storage device 230 stores a security database 240, which includes security information for determining whether a received Downloadable is to be deemed suspicious. The data storage device 230 further stores a users list 260 identifying the users within the internal computer network 115 who may receive Downloadables, and an event log 245 which includes determination results for each Downloadable examined and runtime indications of the internal network security system 110. An operating system 250 controls processing by CPU 205, and is typically stored in data storage device 230 and loaded into RAM 235 (as illustrated) for execution. A security program 255 controls examination of incoming Downloadables, and also may be stored in data storage device 230 and loaded into RAM 235 (as illustrated) for execution by CPU 205.

FIG. 3 is a block diagram illustrating details of the security program 255 and the security database 240. The security program 255 includes an ID generator 315, a policy finder 317 coupled to the ID generator 315, and a first comparator 320 coupled to the policy finder 317. The first comparator 320 is coupled to a logical engine 333 via four separate paths, namely, via Path 1, via Path 2, via Path 3 and via Path 4. Path 1 includes a direct connection from the first comparator 320 to the logical engine 333. Path 2 includes a code scanner coupled to the first comparator 320, and an Access Control List (ACL) comparator 330 coupling the code scanner 325 to the logical engine 333. Path 3 includes a certificate scanner 340 coupled to the first comparator 320, and a certificate comparator 345 coupling the certificate scanner 340 to the logical engine 333. Path 4 includes a Uniform Resource Locator (URL) comparator 350 coupling the first comparator 320 to the logical engine 3330. A record-keeping engine 335 is coupled between the logical engine 333 and the event log 245.

The security program 255 operates in conjunction with the security database 240, which includes security policies 305, known Downloadables 307, known Certificates 309 and Downloadable Security Profile (DSP) data 310 corresponding to the known Downloadables 307. Security policies 305 includes policies specific to particular users 260 and default (or generic) policies for determining whether to allow or block an incoming Downloadable. These security policies 305 may identify specific Downloadables to block, specific Downloadables to allow, or necessary criteria for allowing an unknown Downloadable. Referring to FIG. 4, security policies 305 include policy selectors 405, access control lists 410, trusted certificate lists 415, URL rule bases 420, and lists 425 of Downloadables to allow or to block per administrative override.

Known Downloadables 307 include lists of Downloadables which Original Equipment Manufacturers (OEMs) know to be hostile, of Downloadables which OEMs know to be non-hostile, and of Downloadables previously received by this security program 255. DSP data 310 includes the list of all potentially hostile or suspicious computer operations that may be attempted by each known Downloadable 307, and may also include the respective arguments of these operations. An identified argument of an operation is referred to as "resolved." An unidentified argument is referred to as "unresolved." DSP data 310 is described below with reference to the code scanner 325.

The ID generator 315 receives a Downloadable (including the URL from which it came and the userID of the intended recipient) from the external computer network 105 via the external communications interface 210, and generates a Downloadable ID for identifying each Downloadable. The Downloadable ID preferably includes a digital hash of the complete Downloadable code. The ID generator 315 preferably prefetches all components embodied in or identified by the code for Downloadable ID generation. For example, the ID generator 315 may prefetch all classes embodied in or identified by the Java™ applet bytecode to generate the Downloadable ID. Similarly, the ID generator 315 may retrieve all components listed in the .INF file for an ActiveX™ control to compute a Downloadable ID. Accordingly, the Downloadable ID for the Downloadable will be the same each time the ID generator 315 receives the same Downloadable. The ID generator 315 adds the generated Downloadable ID to the list of known Downloadables 307 (if it is not already listed). The ID generator 315 then forwards the Downloadable and Downloadable ID to the policy finder 317.

The policy finder 317 uses the userID of the intended user and the Downloadable ID to select the specific security policy 305 that shall be applied on the received Downloadable. If there is a specific policy 305 that was defined for the user (or for one of its super groups) and the Downloadable, then the policy is selected. Otherwise the generic policy 305

that was defined for the user (or for one of its super groups) is selected. The policy finder 317 then sends the policy to the first comparator 320.

The first comparator 320 receives the Downloadable, the Downloadable ID and the security policy 305 from the policy finder 317. The first comparator 320 examines the security policy 305 to determine which steps are needed for allowing the Downloadable. For example, the security policy 305 may indicate that, in order to allow this Downloadable, it must pass all four paths, Path 1, Path 2, Path 3 and Path 4. Alternatively, the security policy 305 may indicate that to allow the Downloadable, it must pass only one of the paths. The first comparator 320 responds by forwarding the proper information to the paths identified by the security policy 305.

Path 1

In path 1, the first comparator 320 checks the policy selector 405 of the security policy 305 that was received from the policy finder 317. If the policy selector 405 is either "Allowed" or "Blocked," then the first comparator 320 forwards this result directly to the logical engine 333. Otherwise, the first comparator 320 invokes the comparisons in path 2 and/or path 3 and/or path 4 based on the contents of policy selector 405. It will be appreciated that the first comparator 320 itself compares the Downloadable ID against the lists of Downloadables to allow or block per administrative override 425. That is, the system security administrator can define specific Downloadables as "Allowed" or "Blocked."

Alternatively, the logical engine 333 may receive the results of each of the paths and based on the policy selector 405 may institute the final determination whether to allow or block the Downloadable. The first comparator 320 informs the logical engine 333 of the results of its comparison.

Path 2

In path 2, the first comparator 320 delivers the Downloadable, the Downloadable ID and the security policy 305 to the code scanner 325. If the DSP data 310 of the received Downloadable is known, the code scanner 325 retrieves and forwards the information to the ACL comparator 330. Otherwise, the code scanner 325 resolves the DSP data 310. That is, the code scanner 325 uses conventional parsing techniques to decompose the code (including all prefetched components) of the Downloadable into the DSP data 310. DSP data 310 includes the list of all potentially hostile or suspicious computer operations that may be attempted by a specific Downloadable 307, and may also include the respective arguments of these operations. For example, DSP data 310 may include a READ from a specific file, a SEND to an unresolved host, etc. The code scanner 325 may generate the DSP data 310 as a list of all operations in the Downloadable code which could ever be deemed potentially hostile and a list of all files to be accessed by the Downloadable code. It will be appreciated that the code scanner 325 may search the code for any pattern, which is undesirable or suggests that the code was written by a hacker.

An Example List of Operations Deemed Potentially Hostile

File operations: READ a file, WRITE a file;

Network operations: LISTEN on a socket, CONNECT to a socket, SEND data, RECEIVE data, VIEW INTRANET;

Registry operations: READ a registry item, WRITE a registry item;

Operating system operations: EXIT WINDOWS, EXIT BROWSER, START PROCESS/THREAD, KILL PROCESS/THREAD, CHANGE PROCESS/THREAD PRIORITY, DYNAMICALLY LOAD A CLASS/LIBRARY, etc.; and

Resource usage thresholds: memory, CPU, graphics, etc.

In the preferred embodiment, the code scanner 325 performs a full-content inspection. However, for improved speed but reduced security, the code scanner 325 may examine only a portion of the Downloadable such as the Downloadable header. The code scanner 325 then stores the DSP data into DSP data 310 (corresponding to its Downloadable ID), and sends the Downloadable, the DSP data to the ACL comparator 330 for comparison with the security policy 305.

The ACL comparator 330 receives the Downloadable, the corresponding DSP data and the security policy 305 from the code scanner 325, and compares the DSP data against the security policy 305. That is, the ACL comparator 330 compares the DSP data of the received Downloadable against the access control lists 410 in the received security policy 305. The access control list 410 contains criteria indicating whether to pass or fail the Downloadable. For example, an access control list may indicate that the Downloadable fails if the DSP data includes a WRITE command to a system file. The ACL comparator 330 sends its results to the logical engine 333.

Path 3

In path 3, the certificate scanner 340 determines whether the received Downloadable was signed by a certificate authority, such as VeriSign, Inc., and scans for a certificate embodied in the Downloadable. The certificate scanner 340 forwards the found certificate to the certificate comparator 345. The certificate comparator 345 retrieves known certificates 309 that were deemed trustworthy by the security administrator and compares the found certificate with the known certificates 309 to determine whether the Downloadable was signed by a trusted certificate. The certificate comparator 345 sends the results to the logical engine 333.

Path 4

In path 4, the URL comparator 350 examines the URL identifying the source of the Downloadable against URLs stored in the URL rule base 420 to determine whether the Downloadable comes from a trusted source. Based on the security policy 305, the URL comparator 350 may deem the Downloadable suspicious if the Downloadable comes from an untrustworthy source or if the Downloadable did not come from a trusted source. For example, if the Downloadable comes from a known hacker, then the Downloadable may be deemed suspicious and presumed hostile. The URL comparator 350 sends its results to the logical engine 333.

The logical engine 333 examines the results of each of the paths and the policy selector 405 in the security policy 305 to determine whether to allow or block the Downloadable. The policy selector 405 includes a logical expression of the results received from each of the paths. For example, the logical engine 333 may block a Downloadable if it fails any one of the paths, i.e., if the Downloadable is known hostile (Path 1), if the Downloadable may request suspicious operations (Path 2), if the Downloadable was not signed by a trusted certificate authority (Path 3), or if the Downloadable came from an untrustworthy source (Path 4). The logical engine 333 may apply other logical expressions according to the policy selector 405 embodied in the security policy 305. If the policy selector 405 indicates that the Downloadable may pass, then the logical engine 333 passes the Downloadable to its intended recipient. Otherwise, if the policy selector 405 indicates that the Downloadable should be blocked, then the logical engine 333 forwards a non-hostile Downloadable to the intended recipient to inform the user

that internal network security system 110 discarded the original Downloadable. Further, the logical engine 333 forwards a status report to the record-keeping engine 335, which stores the reports in event log 245 in the data storage device 230 for subsequent review, for example, by the MIS director.

FIG. 5 is a block diagram illustrating details of the security management console 120, which includes a security policy editor 505 coupled to the communications channel 135, an event log analysis engine 510 coupled between communications channel 135 and a user notification engine 515, and a Downloadable database review engine 520 coupled to the communications channel 135. The security management console 120 further includes computer components similar to the computer components illustrated in FIG. 2.

The security policy editor 505 uses an I/O interface similar to I/O interface 215 for enabling authorized user modification of the security policies 305. That is, the security policy editor 505 enables the authorized user to modify specific security policies 305 corresponding to the users 260, the default or generic security policy 305, the Downloadables to block per administrative override, the Downloadables to allow per administrative override, the trusted certificate lists 415, the policy selectors 405, the access control lists 410, the URLs in the URL rule bases 420, etc. For example, if the authorized user learns of a new hostile Downloadable, then the user can add the Downloadable to the Downloadables to block per system override.

The event log analysis engine 510 examines the status reports contained in the event log 245 stored in the data storage device 230. The event log analysis engine 510 determines whether notification of the user (e.g., the security system manager or MIS director) is warranted. For example, the event log analysis engine 510 may warrant user notification whenever ten (10) suspicious Downloadables have been discarded by internal network security system 110 within a thirty (30) minute period, thereby flagging a potential imminent security threat. Accordingly, the event log analysis engine 510 instructs the user notification engine 515 to inform the user. The user notification engine 515 may send an e-mail via internal communications interface 220 or via external communications interface 210 to the user, or may display a message on the user's display device (not shown).

FIG. 6A is a flowchart illustrating a method 600 for protecting an internal computer network 115 from suspicious Downloadables. Method 600 begins with the ID generator 315 in step 602 receiving a Downloadable. The ID generator 315 in step 604 generates a Downloadable ID identifying the received Downloadable, preferably, by generating a digital hash of the Downloadable code (including prefetched components). The policy finder 317 in step 606 finds the appropriate security policy 305 corresponding to the userID specifying intended recipient (or the group to which the intended recipient belongs) and the Downloadable. The selected security policy 305 may be the default security policy 305. Step 606 is described in greater detail below with reference to FIG. 6B.

The first comparator 320 in step 608 examines the lists of Downloadables to allow or to block per administrative override 425 against the Downloadable ID of the incoming Downloadable to determine whether to allow the Downloadable automatically. If so, then in step 612 the first comparator 320 sends the results to the logical engine 333. If not, then the method 600 proceeds to step 610. In step 610, the first comparator 620 examines the lists of Downloadables to block per administrative override 425 against the Downloadable ID of the incoming Downloadable for determining whether to block the Downloadable automatically. If so, then the first comparator 420 in step 612 sends the results to the logical engine 333. Otherwise, method 600 proceeds to step 614.

In step 614, the first comparator 320 determines whether the security policy 305 indicates that the Downloadable should be tested according to Path 4. If not, then method 600 jumps to step 618. If so, then the URL comparator 350 in step 616 compares the URL embodied in the incoming Downloadable against the URLs of the URL rules bases 420, and then method 600 proceeds to step 618.

In step 618, the first comparator 320 determines whether the security policy 305 indicates that the Downloadable should be tested according to Path 2. If not, then method 600 jumps to step 620. Otherwise, the code scanner 235 in step 626 examines the DSP data 310 based on the Downloadable ID of the incoming Downloadable to determine whether the Downloadable has been previously decomposed. If so, then method 600 jumps to step 630. Otherwise, the code scanner 325 in step 628 decomposes the Downloadable into DSP data. Downloadable decomposition is described in greater detail with reference to FIG. 7. In step 630, the ACL comparator 330 compares the DSP data of the incoming Downloadable against the access control lists 410 (which include the criteria necessary for the Downloadable to fail or pass the test).

In step 620, the first comparator 320 determines whether the security policy 305 indicates that the Downloadable should be tested according to Path 3. If not, then method 600 returns to step 612 to send the results of each of the test performed to the logical engine 333. Otherwise, the certificate scanner 622 in step 622 scans the Downloadable for an embodied certificate. The certificate comparator 345 in step 624 retrieves trusted certificates from the trusted certificate lists (TCL) 415 and compares the embodied certificate with the trusted certificates to determine whether the Downloadable has been signed by a trusted source. Method 600 then proceeds to step 612 by the certificate scanner 345 sending the results of each of the paths taken to the logical engine 333. The operations of the logical engine 333 are described in greater detail below with reference to FIG. 6C. Method 600 then ends.

One skilled in the art will recognize that the tests may be performed in a different order, and that each of the tests need not be performed. Further, one skilled in the art will recognize that, although path 1 is described in FIG. 6A as an automatic allowance or blocking, the results of Path 1 may be another predicate to be applied by the logical engine 333. Further, although the tests are shown serially in FIG. 6A, the tests may be performed in parallel as illustrated in FIG. 3.

FIG. 6B is a flowchart illustrating details of step 606 of FIG. 6A (referred to herein as method 606). Method 606 begins with the policy finder 317 in step 650 determining whether security policies 305 include a specific security policy corresponding to the userID and the Downloadable. If so, then the policy finder 317 in step 654 fetches the corresponding specific policy 305. If not, then the policy finder 317 in step 652 fetches the default or generic security policy 305 corresponding to the userID. Method 606 then ends.

FIG. 6C is a flowchart illustrating details of a method 655 for determining whether to allow or to block the incoming Downloadable. Method 655 begins with the logical engine 333 in step 660 receiving the results from the first comparator 320, from the ACL comparator 330, from the certificate

comparator 345 and from the URL comparator 350. The logical engine 333 in step 662 compares the results with the policy selector 405 embodied in the security policy 305, and in step 664 determines whether the policy selector 405 confirms the pass. For example, the policy selector 405 may indicate that the logical engine 333 pass the Downloadable if it passes one of the tests of Path 1, Path 2, Path 3 and Path 4. If the policy selector 405 indicates that the Downloadable should pass, then the logical engine 333 in step 666 passes the Downloadable to the intended recipient. In step 668, the logical engine 333 sends the results to the record-keeping engine 335, which in turn stores the results in the event log 245 for future review. Method 655 then ends. Otherwise, if the policy selector 405 in step 664 indicates that the Downloadable should not pass, then the logical engine 333 in step 670 stops the Downloadable and in step 672 sends a non-hostile substitute Downloadable to inform the user that the incoming Downloadable has been blocked. Method 655 then jumps to step 668.

FIG. 7 is a flowchart illustrating details of step 628 of FIG. 6A (referred to herein as method 628) for decomposing a Downloadable into DSP data 310. Method 628 begins in step 705 with the code scanner 325 disassembling the machine code of the Downloadable. The code scanner 325 in step 710 resolves a respective command in the machine code, and in step 715 determines whether the resolved command is suspicious (e.g., whether the command is one of the operations identified in the list described above with reference to FIG. 3). If not, then the code scanner 325 in step 725 determines whether it has completed decomposition of the Downloadable, i.e., whether all operations in the Downloadable code have been resolved. If so, then method 628 ends. Otherwise, method 628 returns to step 710.

Otherwise, if the code scanner 325 in step 715 determines that the resolved command is suspect, then the code scanner 325 in step 720 decodes and registers the suspicious command and its command parameters as DSP data 310. The code scanner 325 in step 720 registers the commands and command parameters into a format based on command class (e.g., file operations, network operations, registry operations, operating system operations, resource usage thresholds). Method 628 then jumps to step 725.

FIG. 8 is a flowchart illustrating a method 800 for generating a Downloadable ID for identifying a Downloadable. Method 800 begins with the ID generator 315 in step 810 receiving a Downloadable from the external computer network 105. The ID generator 315 in step 820 may fetch some or all components referenced in the Downloadable code, and in step 830 includes the fetched components in the Downloadable code. The ID generator 315 in step 840 performs a hashing function on at least a portion of the Downloadable code to generate a Downloadable ID. The ID generator 315 in step 850 stores the generated Downloadable ID in the security database 240 as a reference to the DSP data 310. Accordingly, the Downloadable ID will be the same for the identical Downloadable each time it is encountered.

The foregoing description of the preferred embodiments of the invention is by way of example only, and other variations of the above-described embodiments and methods are provided by the present invention. For example, although the invention has been described in a system for protecting an internal computer network, the invention can be embodied in a system for protecting an individual computer. Components of this invention may be implemented using a programmed general purpose digital computer, using application specific integrated circuits, or using a network of interconnected conventional components and circuits. The embodiments described herein have been presented for purposes of illustration and are not intended to be exhaustive or limiting. Many variations and modifications are possible in light of the foregoing teaching. The system is limited only by the following claims.

What is claimed is:

1. A computer-based method, comprising the steps of:

receiving an incoming Downloadable addressed to a client, by a server that serves as a gateway to the client;

comparing, by the server, Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against a security policy to determine if the security policy has been violated; and

preventing execution of the Downloadable by the client if the security policy has been violated.

2. The method of claim 1, further comprising the step of decomposing the Downloadable into the Downloadable security profile data.

3. The method of claim 2, wherein the security policy includes an access control list and further comprising the step of comparing the Downloadable security profile data against the access control list.

4. The method of claim 1, further comprising the steps of scanning for a certificate and comparing the certificate against a trusted certificate.

5. The method of claim 1, further comprising the step of comparing the URL from which the Downloadable originated against a known URL.

6. The method of claim 5, wherein the known URL is a trusted URL.

7. The method of claim 5, wherein the known URL is an untrusted URL.

8. The method of claim 1, wherein the Downloadable includes a Java™ applet.

9. The method of claim 1, wherein the Downloadable includes an ActiveX™ control.

10. The method of claim 1, wherein the Downloadable includes a JavaScript™ script.

11. The method of claim 1, wherein the Downloadable includes a Visual Basic script.

12. The method of claim 1, wherein

the security policy includes a default security policy to be applied regardless of the client to whom the Downloadable is addressed.

13. The method of claim 1, wherein

the security policy includes a specific security policy corresponding to the client to whom the Downloadable is addressed.

14. The method of claim 1, wherein

the client belongs to a particular group; and

the security policy includes a specific security policy corresponding to the particular group.

15. The method of claim 1, further comprising, after preventing execution of the Downloadable, the step of sending a substitute non-hostile Downloadable to the client for informing the client.

16. The method of claim 1, further comprising, after preventing execution of the Downloadable, the step of recording the violation in an event log.

17. The method of claim 1, further comprising the step of computing a Downloadable ID to identify the Downloadable.

18. The method of claim 16, further comprising the steps of fetching components identified by the Downloadable and including the fetched components in the Downloadable.

19. The method of claim 18, further comprising the step of performing a hashing function on the Downloadable to compute a Downloadable ID to identify the Downloadable.

20. The method of claim 18, further comprising the step of fetching all components identified by the Downloadable.

21. The method of claim 1 further comprising the step of examining the intended recipient userID to determine the appropriate security policy.

22. The method of claim 20, wherein the appropriate security policy includes a default security policy.

23. The method of claim 1, further comprising the step of examining the Downloadable to determine the appropriate security policy.

24. The method of claim 1, further comprising the step of comparing the Downloadable against a known Downloadable.

25. The method of claim 24, wherein the known Downloadable is hostile.

26. The method of claim 24, wherein the known Downloadable is non-hostile.

27. The method of claim 24, further comprising the step of including a previously received Downloadable as a known Downloadable.

28. The method of claim 27, wherein the security policy identifies a Downloadable to be blocked per administrative override.

29. The method of claim 28, wherein the security policy identifies a Downloadable to be allowed per administrative override.

30. The method of claim 1, further comprising the step of informing a user upon detection of a security policy violation.

31. The method of claim 1, further comprising the steps of recognizing the incoming Downloadable, and obtaining the Downloadable security profile data for the incoming Downloadable from memory.

32. A system for execution by a server that serves as a gateway to a client, the system comprising:

a security policy;

an interface for receiving an incoming Downloadable addressed to a client;

a comparator, coupled to the interface, for comparing Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against the security policy to determine if the security policy has been violated; and

a logical engine for preventing execution of the Downloadable by the client if the security policy has been violated.

33. The system of claim 32, wherein the Downloadable includes a Java™ applet.

34. The system of claim 32, wherein the Downloadable includes ActiveX™ control.

35. The system of claim 32, wherein the Downloadable includes a JavaScript™ script.

36. The system of claim 32, wherein the Downloadable includes a Visual Basic script.

37. The system of claim 32, wherein

the security policy includes a default security policy to be applied regardless of the client to whom the Downloadable is addressed.

38. The system of claim 32, wherein

the security policy includes a specific security policy corresponding to the client to whom the Downloadable is addressed.

39. The system of claim 32, wherein

the client belongs to a particular group; and

the security policy includes a specific security policy corresponding to the particular group.

40. The system of claim 32, further comprising an ID generator coupled to the interface for computing a Downloadable ID identifying the Downloadable.

41. The system of claim 40, wherein the ID generator prefetches all components of the Downloadable and uses all components to compute the Downloadable ID.

42. The system of claim 41, wherein the ID generator computes the digital hash of all the prefetched components.

43. The system of claim 32, further comprising a policy finder for finding the security policy.

44. The system of claim 43, wherein the policy finder finds the security policy based on the user.

45. The system of claim 43 wherein the policy finder finds the security policy based on the user and the Downloadable.

46. The system of claim 43, wherein the policy finder obtains the default security policy.

47. The system of claim 32 wherein the comparator examines the security policy to determine which tests to apply.

48. The system of claim 47 wherein the comparator compares the Downloadable against a known Downloadable.

49. The system of claim 48, wherein the known Downloadable is hostile.

50. The system of claim 48, wherein the known Downloadable is non-hostile.

51. The system of claim 32, wherein the security policy identifies a Downloadable to be blocked per administrative override.

52. The system of claim 32, wherein the security policy identifies a Downloadable to be allowed per administrative override.

53. The system of claim 32, wherein

the comparator sends a substitute non-hostile Downloadable to the client for informing the client.

54. The system of claim 32, further comprising a code scanner coupled to the comparator for decomposing the Downloadable into the Downloadable security profile data.

55. The system of claim 54, further comprising an ACL comparator coupled to the code scanner for comparing the Downloadable security profile data against an access control list.

56. The system of claim 32, further comprising a certificate scanner coupled to the comparator for examining the Downloadable for a certificate.

57. The system of claim 56, further comprising a certificate comparator coupled to the certificate scanner for comparing the certificate against a trusted certificate.

58. The system of claim 32, further comprising a URL comparator coupled to the comparator for comparing the URL from which the Downloadable originated against a known URL.

59. The system of claim 58, wherein the known URL identifies an untrusted URL.

60. The system of claim 58, wherein the known URL identifies a trusted URL.

61. The system of claim 31, wherein the logical engine responds according to the security policy.

62. The system of claim 31, further comprising a record-keeping engine coupled to the comparator for recording results in an event log.

63. The system of claim 32, further comprising memory storing the Downloadable security profile data for the incoming Downloadable.

64. A system for execution on a server that serves as a gateway to a client, comprising:

means for receiving an incoming Downloadable addressed to a client;

means for comparing Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against a security policy to determine if the security policy has been violated; and

means for preventing execution of the Downloadable by the client if the security policy has been violated.

65. A computer-readable storage medium storing program code for causing a server that serves as a gateway to a client to perform the steps of:

receiving an incoming Downloadable addressed to a client;

comparing Downloadable security profile data pertaining to the Downloadable against a security policy to determine if the security policy has been violated; and

preventing execution of the Downloadable by the client if the security policy has been violated.

66. A method, comprising:

receiving a Downloadable;

decomposing the Downloadable into Downloadable security profile data; the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable,

comparing the Downloadable security profile data against a security policy; and

preventing execution of the Downloadable if the Downloadable security profile data violates the security policy.

67. The method of claim 66, further comprising:

fetching all components referenced by the Downloadable;

performing a hashing function of the Downloadable and the components fetched to compute a Downloadable ID; and

storing the Downloadable security profile data and the Downloadable ID in memory.

68. A method, comprising:

providing memory storing known-Downloadable security profile data and a that includes a list a suspicious computer operations that may be attempted by a Downloadable known-Downloadable ID corresponding to the Downloadable security profile data;

receiving an incoming Downloadable;

fetching all components referenced by the incoming Downloadable;

performing a hashing function of the Downloadable and the components to compute an incoming-Downloadable ID;

comparing the known-Downloadable ID against the incoming-Downloadable ID;

retrieving the Downloadable security profile data if the known-Downloadable ID and the incoming-Downloadable ID match; and

comparing the Downloadable security profile data against a security policy to determine if the incoming Downloadable violates the security policy.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 6,092,194
DATED : July 18, 2000
INVENTOR(S) : Shlomo Touboul

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Column 13,
Line 19, after "to the Downloadable" and before "against a security" insert –, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, –

Column 14,
Line 12, after "profile data and" and before "that includes" delete – a –

Signed and Sealed this

Fifth Day of February, 2002

Attest:

Attesting Officer

JAMES E. ROGAN
Director of the United States Patent and Trademark Office

US006804780B1

(12) **United States Patent**
Touboul

(10) **Patent No.: US 6,804,780 B1**
(45) **Date of Patent: *Oct. 12, 2004**

(54) **SYSTEM AND METHOD FOR PROTECTING A COMPUTER AND A NETWORK FROM HOSTILE DOWNLOADABLES**

(75) Inventor: Shlomo Touboul, Kefar-haim (IL)

(73) Assignee: Finjan Software, Ltd., Netanya (IL)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: 09/539,667

(22) Filed: Mar. 30, 2000

Related U.S. Application Data

(63) Continuation of application No. 08/964,388, filed on Nov. 6, 1997, now Pat. No. 6,092,194.
(60) Provisional application No. 60/030,639, filed on Nov. 8, 1996.

(51) Int. Cl.[7] H04L 9/00; G06F 11/30
(52) U.S. Cl. 713/181; 713/201; 713/176; 717/178
(58) Field of Search 713/200, 201, 713/176, 181; 709/223, 225, 227, 229; 717/168–178

(56) References Cited

U.S. PATENT DOCUMENTS

5,077,677	A	12/1991	Murphy et al.	
5,359,659	A	10/1994	Rosenthal	
5,361,359	A	11/1994	Tajalli et al.	
5,485,409	A	1/1996	Gupta et al.	
5,485,575	A	1/1996	Chess et al.	
5,572,643	A	11/1996	Judson	
5,579,509	A	* 11/1996	Furtney et al.	703/27
5,606,668	A	2/1997	Shwed	
5,623,600	A	4/1997	Ji et al.	
5,638,446	A	6/1997	Rubin	
5,692,047	A	11/1997	McManis	
5,692,124	A	11/1997	Holden et al.	
5,720,033	A	2/1998	Deo	
5,724,425	A	3/1998	Chang et al.	
5,740,248	A	4/1998	Fieres et al.	
5,761,421	A	6/1998	van Hoff et al.	

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

EP	1091276	A1 *	4/2001	G06F/1/00
EP	1132796	A1 *	9/2001	G06F/1/00

OTHER PUBLICATIONS

Khare, "Microsoft Authenticode Analyzed" Jul. 22, 1996, xent.com/FoRK-archive/summer96/0338.html, p. 1–2.*

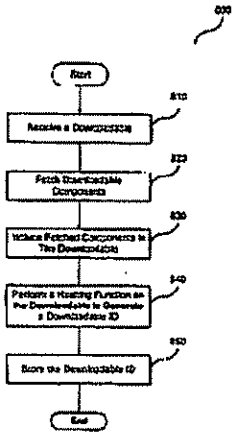
(List continued on next page.)

Primary Examiner—Ayaz Sheikh
Assistant Examiner—Christopher Revak
(74) *Attorney, Agent, or Firm*—Squire, Sanders & Dempsey, L.L.P.

(57) **ABSTRACT**

A computer-based method for generating a Downloadable ID to identify a Downloadable, including obtaining a Downloadable that includes one or more references to software components required by the Downloadable, fetching at least one software component identified by the one or more references, and performing a function on the Downloadable and the fetched software components to generate a Downloadable ID. A system and a computer-readable storage medium are also described and claimed.

18 Claims, 10 Drawing Sheets



U.S. PATENT DOCUMENTS

5,765,205	A		6/1998	Breslau et al.
5,784,459	A		7/1998	Devarakonda et al.
5,796,952	A		8/1998	Davis et al.
5,805,829	A		9/1998	Cohen et al.
5,832,208	A		11/1998	Chen et al.
5,832,274	A	*	11/1998	Cutler et al. 717/171
5,850,559	A		12/1998	Angelo et al.
5,859,966	A		1/1999	Hayman et al.
5,864,683	A		1/1999	Boebert et al.
5,892,904	A		4/1999	Atkinson et al.
5,951,698	A		9/1999	Chen et al.
5,956,481	A		9/1999	Walsh et al.
5,974,549	A		10/1999	Golan
5,978,484	A	*	11/1999	Apperson et al. 705/54
5,983,348	A		11/1999	Ji
6,092,194	A	*	7/2000	Touboul 713/200
6,154,844	A	*	11/2000	Touboul et al. 713/201
6,339,829	B1	*	1/2002	Beadle et al. 713/201

OTHER PUBLICATIONS

"Release Notes for the Microsfot ActiveX Development Kit", Aug. 13, 1996, activex.adsp.or.jp/inetsdk/readme.txt, p. 1–10.*

"Microsoft ActiveX Software Development Kit" Aug. 12, 1996, activex.adsp.or.jp/inetsdk/help/overview.htm, p. 1–6.*

Doyle et al, "Microsoft Press Computer Dictionary" 1993, Microsoft Press, 2nd Edition, p. 137–138.*

Schmitt, ".EXE. files, OS–2 style" Nov. 1988, PC Tech Journal via dialog search, vol. 6, #11, p. 76–78.*

Jim K. Omura, "Novel Applications of Cryptography in Digital Communications", IEEE Communications Magazine, May, 1990; pp. 21–29.

Okamoto, E. et al., "ID–Based Authentication System For Computer Virus Detection", IEEE/IEE Electronic Library online, Electronics Letters, vol. 26, Issue 15, ISSN 0013/5194, Jul. 19, 1990, Abstract and pp. 1169–1170. URL: http://iel.ihs.com:80/cgi–bin/iel_cgi?hc 2ehis%26ViewTemplate%3ddocview%5fb%2ehis.

IBM AntiVirus User's Guide Version 2.4, International Business Machines Corporation, Nov. 15, 1995, pp. 6–7.

Norvin Leach et al, "IE 3.0 Applets Will Earn Certification", PC Week, vol. 13, No. 29, Jul. 22, 1996, 2 pages.

"Finjan Software Releases SurfinBoard, Industry's First JAVA Security Product For the World Wide Web", Article published on the Internet by Finjan Software Ltd., Jul. 29, 1996, 1 page.

"Powerful PC Security for the New World of Java™ and Downloadables, Surfin Shield™" Article published on the Internet by Finjan Software Ltd., 1996, 2 Pages.

Microsoft® Authenticode Technology, "Ensuring Accountability and Authenticity for Software Components on the Internet", Microsoft Corporation, Oct. 1996, including Abstract, Contents, Introduction and pp. 1–10.

"Finjan Announces a Personal Java™ Firewall For Web Browsers—the SurfinShield™ 1.6 (formerly known as SurfinBoard)", Press Release of Finjan Releases SurfinShield 1.6, Oct. 21, 1996, 2 pages.

Company Profile "Finjan—Safe Surfing, The Java Security Solutions Provider", Article published on the Internet by Finjan Software Ltd., Oct. 31, 1996, 3 pages.

"Finjan Announces Major Power Boost and New Features for SurfinShield™ 2.0" Las Vegas Convention Center/Pavilion 5 PS551, Nov. 18, 1996, 3 pages.

"Java Security: Issues & Solutions" Article published on the Internet by Finjan Software Ltd., 1996, 8 pages.

"Products" Article published on the Internet, 7 pages.

Mark LaDue, "Online Business Consultant: Java Security: Whose Business Is It?" Article published on the Internet, Home Page Press, Inc. 1996, 4 pages.

Web Page Article "Frequently Asked Questions About Authenticode", Microsoft Corporation, last updated Feb. 17, 1997, Printed Dec. 23, 1998. URL: http://www.microsoft.com/workshop/security/authcode/signfaq.asp#9, pp. 1–13.

Zhang, X.N., "Secure Code Distribution", IEEE/IEE Electronic Library online, Computer, vol. 30, Issue 6, Jun., 1997, pp. 76–79.

* cited by examiner

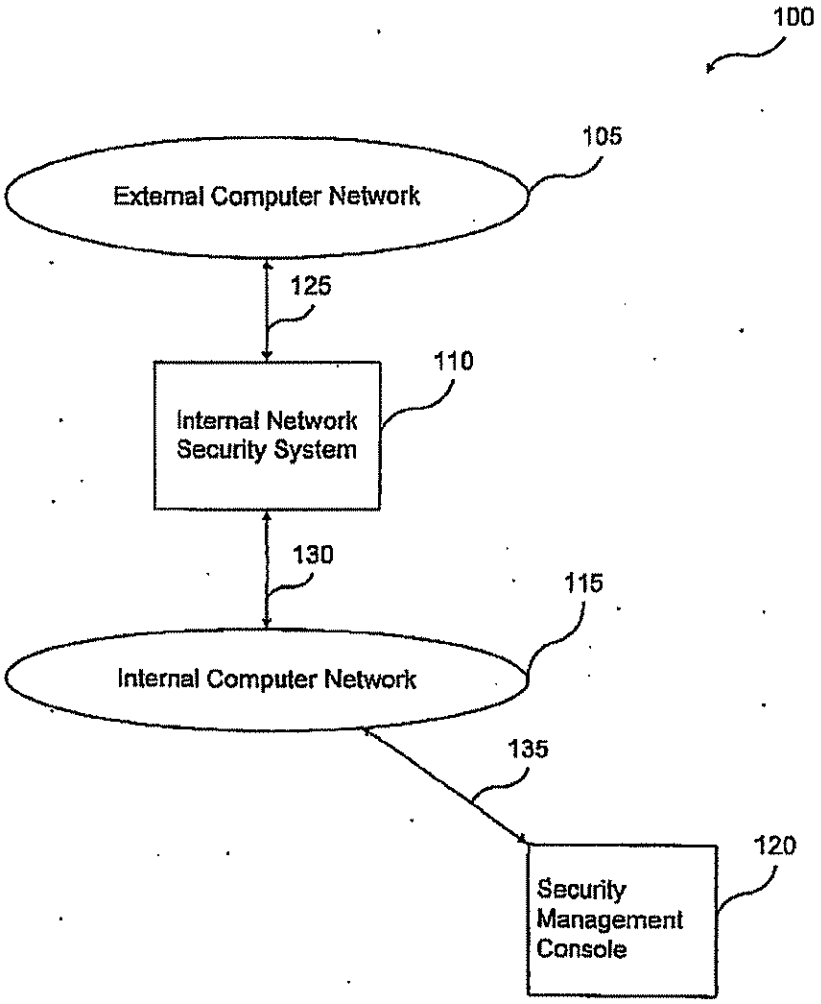

100
105
External Computer Network
125
110
Internal Network Security System
130
115
Internal Computer Network
135
120
Security Management Console

FIG. 1

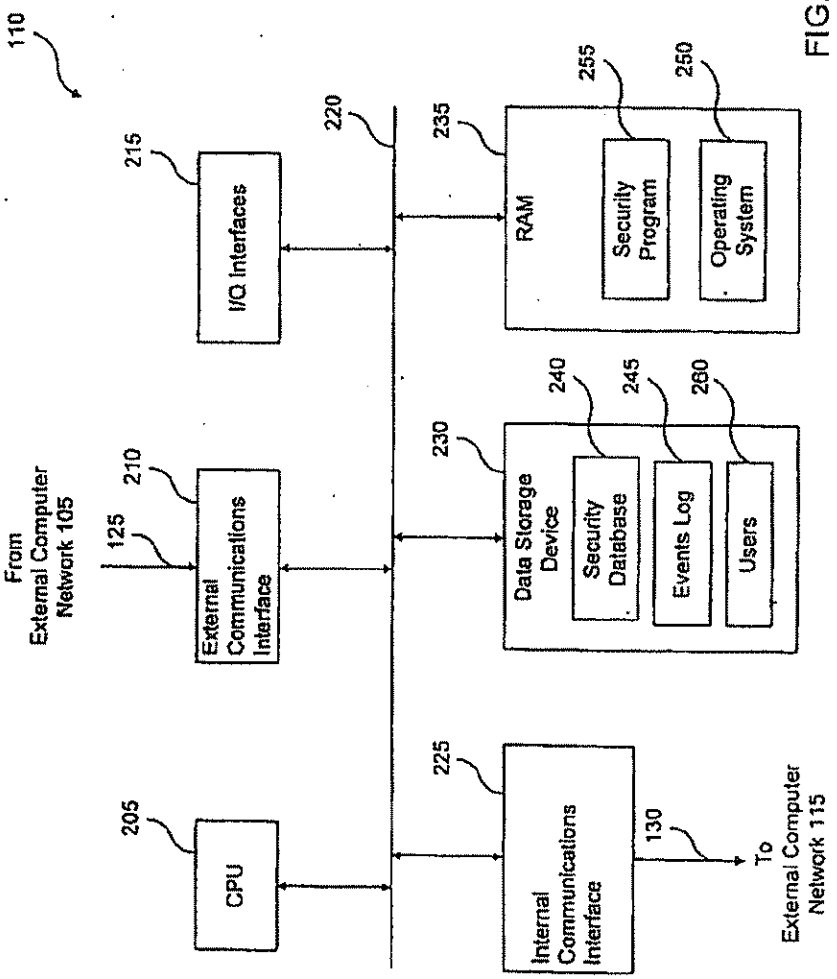

110
215
I/O Interfaces
220
235
RAM
255
Security Program
250
Operating System
From External Computer Network 105
125
210
External Communications Interface
230
Data Storage Device
240
Security Database
245
Events Log
260
Users
205
CPU
225
Internal Communications Interface
130
To External Computer Network 115

FIG. 2

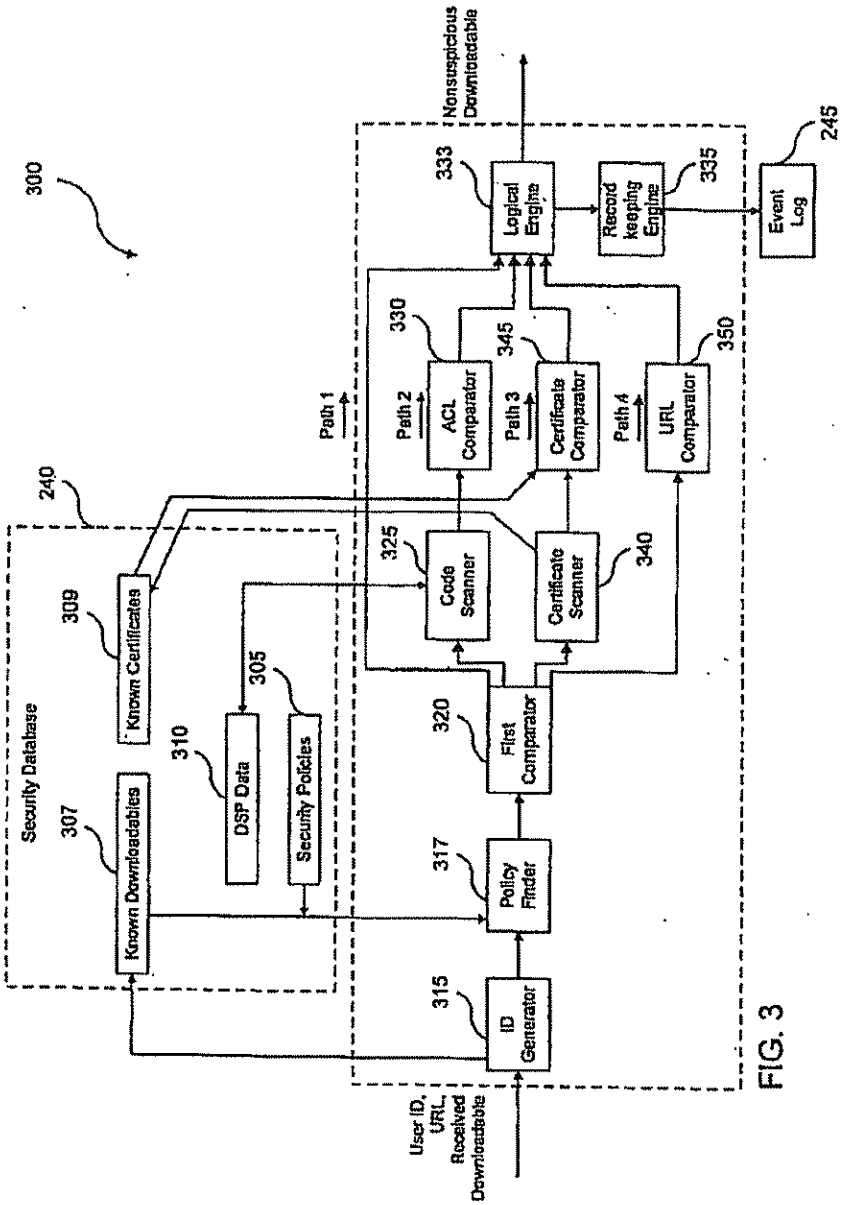

300
Nonsuspicious Downloadable
333
Logical Engine
335
Record keeping Engine
245
Event Log
330
ACL Comparator
345
Certificate Comparator
350
URL Comparator
Path 1
Path 2
Path 3
Path 4
240
325
Code Scanner
340
Certificate Scanner
309
Known Certificates
305
DSP Data
Security Policies
310
320
First Comparator
307
Known Downloadables
317
Policy Finder
315
ID Generator
Security Database
User ID, URL, Received Downloadable

FIG. 3

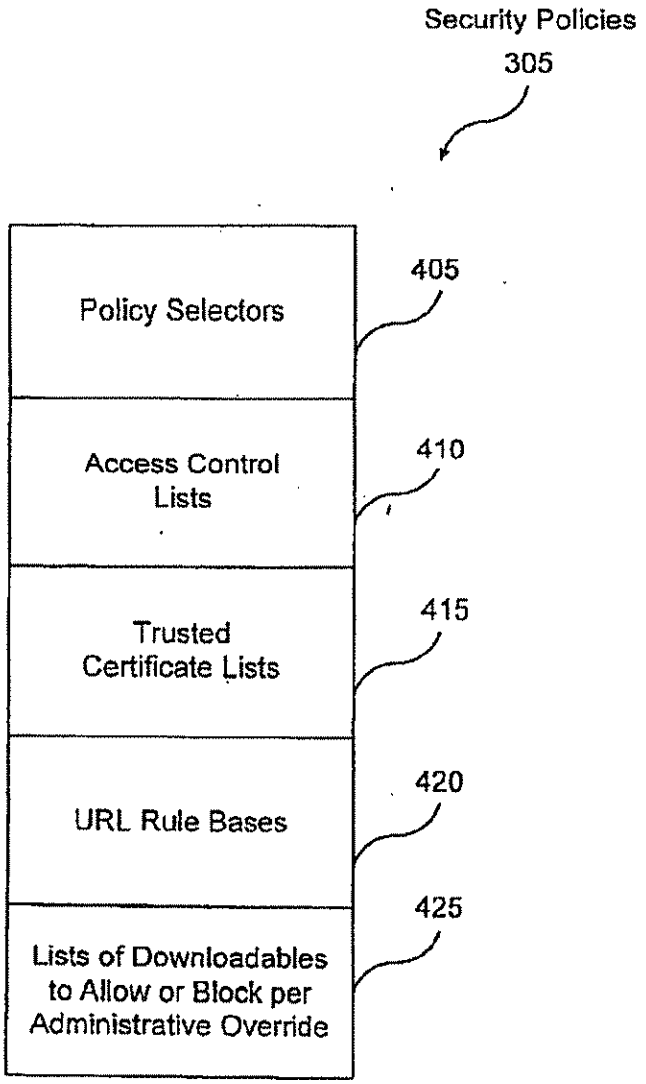

Security Policies
305
Policy Selectors
405
Access Control Lists
410
Trusted Certificate Lists
415
URL Rule Bases
420
Lists of Downloadables to Allow or Block per Administrative Override
425

FIG. 4

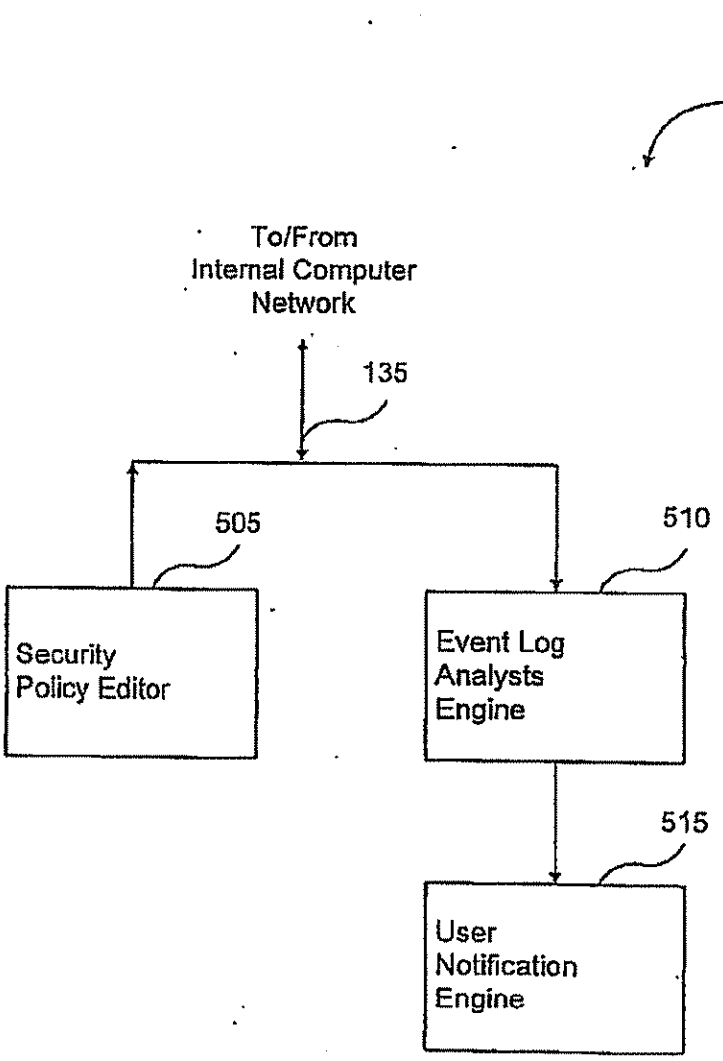

120
To/From
Internal Computer
Network
135
505
510
Security
Policy Editor
Event Log
Analysts
Engine
515
User
Notification
Engine

FIG. 5

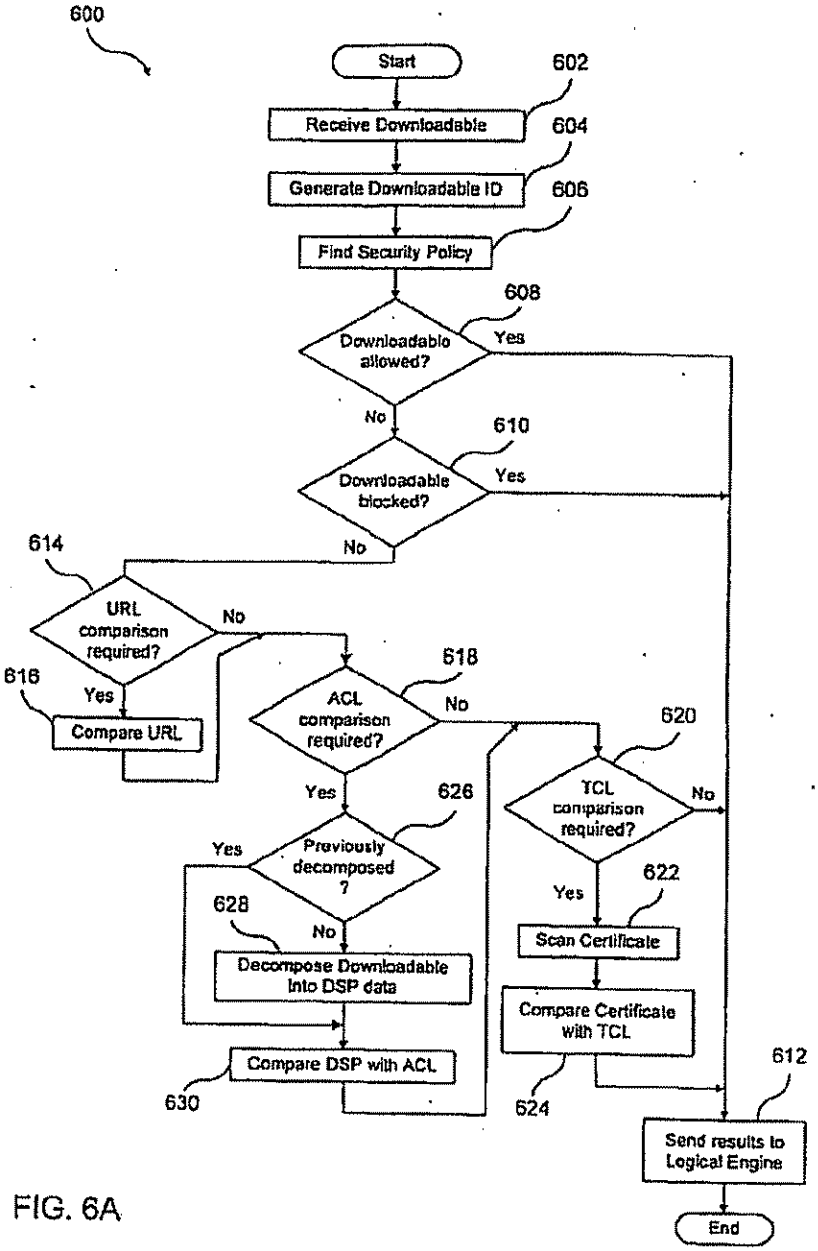

600
Start
602
Receive Downloadable
604
Generate Downloadable ID
606
Find Security Policy
608
Downloadable allowed?
Yes
No
610
Downloadable blocked?
Yes
No
614
URL comparison required?
No
616
Yes
Compare URL
618
ACL comparison required?
No
620
TCL comparison required?
No
Yes
622
Scan Certificate
Compare Certificate with TCL
624
626
Previously decomposed ?
Yes
628
No
Decompose Downloadable into DSP data
Compare DSP with ACL
630
612
Send results to Logical Engine
End

FIG. 6A

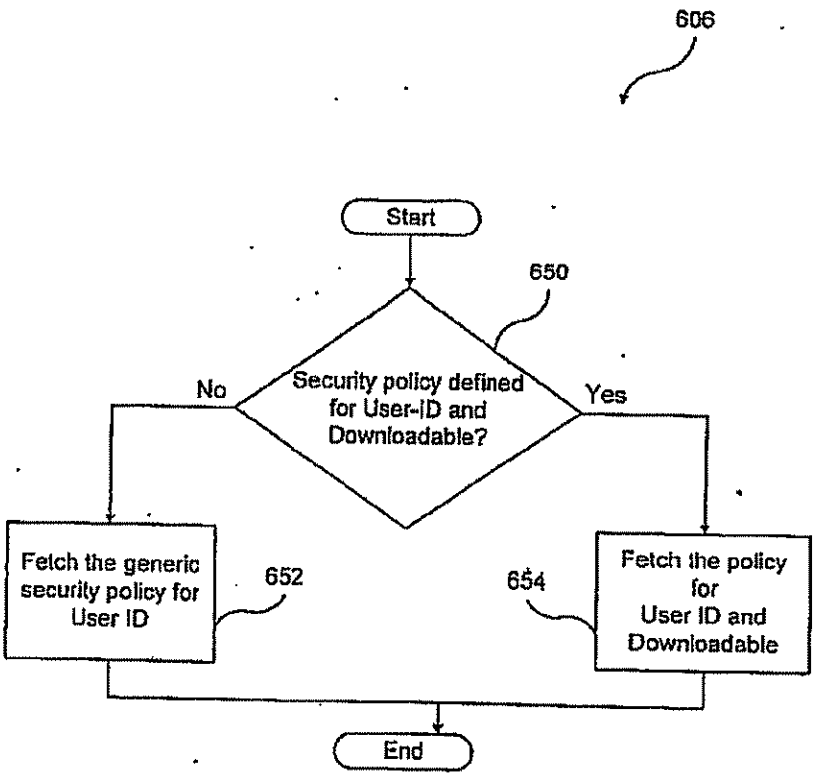

606
Start
650
Security policy defined for User-ID and Downloadable?
No
Yes
Fetch the generic security policy for User ID
652
654
Fetch the policy for User ID and Downloadable
End

FIG. 6B

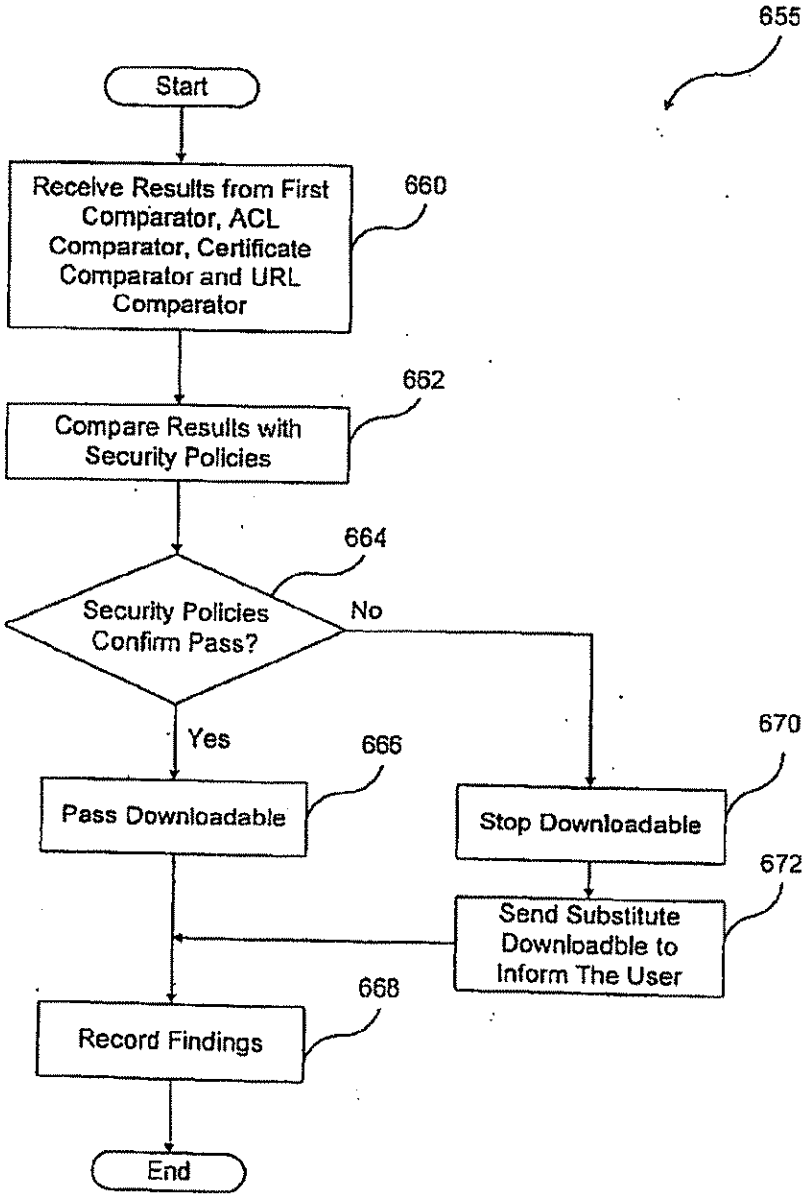

655
Start
Receive Results from First Comparator, ACL Comparator, Certificate Comparator and URL Comparator
660
Compare Results with Security Policies
662
Security Policies Confirm Pass?
664
No
Yes
Pass Downloadable
666
Stop Downloadable
670
Send Substitute Downloadble to Inform The User
672
Record Findings
668
End

FIG. 6C

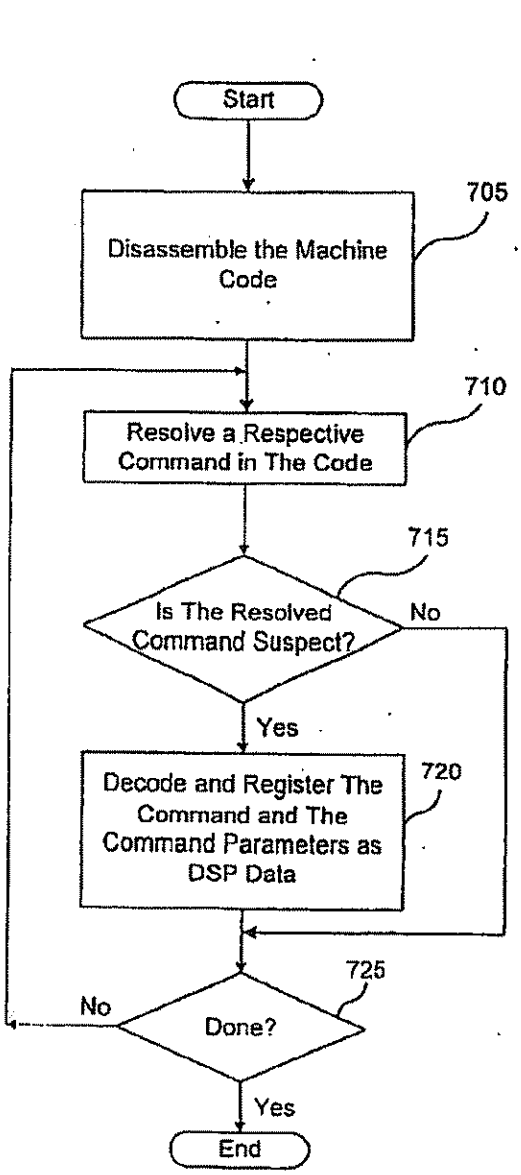

628
Start
705
Disassemble the Machine Code
710
Resolve a Respective Command in The Code
715
Is The Resolved Command Suspect?
No
Yes
720
Decode and Register The Command and The Command Parameters as DSP Data
725
Done?
No
Yes
End

FIG. 7

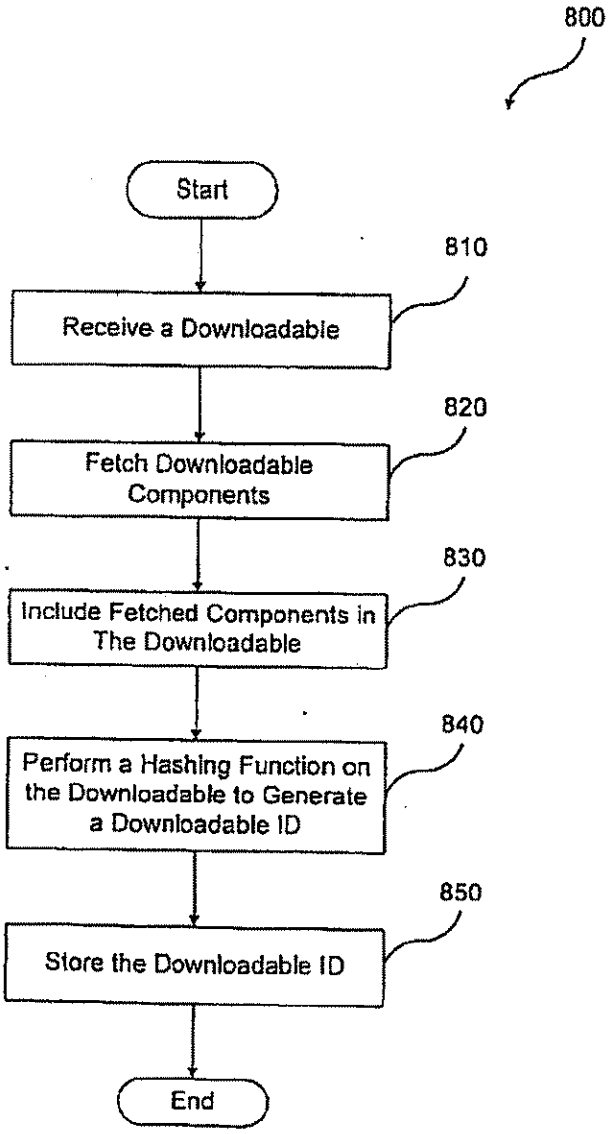

800
Start
810
Receive a Downloadable
820
Fetch Downloadable Components
830
Include Fetched Components in The Downloadable
840
Perform a Hashing Function on the Downloadable to Generate a Downloadable ID
850
Store the Downloadable ID
End

FIG. 8

SYSTEM AND METHOD FOR PROTECTING A COMPUTER AND A NETWORK FROM HOSTILE DOWNLOADABLES

PRIORITY REFERENCE TO RELATED APPLICATION

This application is a continuation of and hereby incorporates by reference U.S. patent application Ser. No. 08/964,388, entitled "System and Method for Protecting a Computer and a Network from Hostile Downloadables," filed Nov. 6, 1997, which is now U.S. Pat. No. 6,092,194, which claims priority to provisional application Serial No. 60/030,639, entitled "System and Method for Protecting a Computer from Hostile Downloadables," filed on Nov. 8, 1996, by inventor Shlomo Touboul.

INCORPORATION BY REFERENCE TO RELATED APPLICATIONS

This application hereby incorporates by reference related U.S. patent application Ser. No. 08/790,097, entitled "System and Method for Protecting a Client from Hostile Downloadables," filed on Jan. 29, 1997, which is now U.S. Pat. No. 6,167,520, by inventor Shlomo Touboul; and hereby incorporates by reference provisional application Ser. No. 60/030,639, entitled "System and Method for Protecting a Computer from Hostile Downloadables," filed on Nov. 8, 1996, by inventor Shlomo Touboul.

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates generally to computer networks, and more particularly provides a system and method for protecting a computer and a network from hostile Downloadables.

2. Description of the Background Art

The Internet is currently a collection of over 100,000 individual computer networks owned by governments, universities, nonprofit groups and companies, and is expanding at an accelerating rate. Because the Internet is public, the Internet has become a major source of many system damaging and system fatal application programs, commonly referred to as "viruses."

Accordingly, programmers continue to design computer and computer network security systems for blocking these viruses from attacking both individual and network computers. On the most part, these security systems have been relatively successful. However, these security systems are not configured to recognize computer viruses which have been attached to or configured as Downloadable application programs, commonly referred to as "Downloadables." A Downloadable is an executable application program, which is downloaded from a source computer and run on the destination computer. Downloadable is typically requested by an ongoing process such as by an Internet browser or web engine. Examples of Downloadables include Java™ applets designed for use in the Java™ distributing environment developed by Sun Microsystems, Inc., JavaScript scripts also developed by Sun Microsystems, Inc., ActiveX™ controls designed for use in the ActiveX™ distributing environment developed by the Microsoft Corporation, and Visual Basic also developed by the Microsoft Corporation. Therefore, a system and method are needed to protect a network from hostile Downloadables.

SUMMARY OF THE INVENTION

The present invention provides a system for protecting a network from suspicious Downloadables. The system comprises a security policy, an interface for receiving a Downloadable, and a comparator, coupled to the interface, for applying the security policy to the Downloadable to determine if the security policy has been violated. The Downloadable may include a Java™ applet, an ActiveX™ control, a JavaScript™ script, or a Visual Basic script. The security policy may include a default security policy to be applied regardless of the client to whom the Downloadable is addressed, a specific security policy to be applied based on the client or the group to which the client belongs, or a specific policy to be applied based on the client/group and on the particular Downloadable received. The system uses an ID generator to compute a Downloadable ID identifying the Downloadable, preferably, by fetching all components of the Downloadable and performing a hashing function on the Downloadable including the fetched components.

Further, the security policy may indicate several tests to perform, including (1) a comparison with known hostile and non-hostile Downloadables; (2) a comparison with Downloadables to be blocked or allowed per administrative override; (3) a comparison of the Downloadable security profile data against access control lists; (4) a comparison of a certificate embodied in the Downloadable against trusted certificates; and (5) a comparison of the URL from which the Downloadable originated against trusted and untrusted URLs. Based on these tests, a logical engine can determine whether to allow or block the Downloadable.

The present invention further provides a method for protecting a computer from suspicious Downloadables. The method comprises the steps of receiving a Downloadable, comparing the Downloadable against a security policy to determine if the security policy has been violated, and discarding the Downloadable if the security policy has been violated.

It will be appreciated that the system and method of the present invention may provide computer protection from known hostile Downloadables. The system and method of the present invention may identify Downloadables that perform operations deemed suspicious. The system and method of the present invention may examine the Downloadable code to determine whether the code contains any suspicious operations, and thus may allow or block the Downloadable accordingly.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram illustrating a network system, in accordance with the present invention;

FIG. 2 is a block diagram illustrating details of the internal network security system of FIG. 1;

FIG. 3 is a block diagram illustrating details of the security program and the security database of FIG. 2;

FIG. 4 is a block diagram illustrating details of the security policies of FIG. 3;

FIG. 5 is a block diagram illustrating details of the security management console of FIG. 1;

FIG. 6A is a flowchart illustrating a method of examining for suspicious Downloadables, in accordance with the present invention;

FIG. 6B is a flowchart illustrating details of the step for finding the appropriate security policy of FIG. 6A;

FIG. 6C is a flowchart illustrating a method for determining whether an incoming Downloadable is to be deemed suspicious;

FIG. 7 is a flowchart illustrating details of the FIG. 6 step of decomposing a Downloadable; and

FIG. 8 is a flowchart illustrating a method 800 for generating a Downloadable ID for identifying a Downloadable.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

FIG. 1 is a block diagram illustrating a network system 100, in accordance with the present invention. The network system 100 includes an external computer network 105, such as the Wide Area Network (WAN) commonly referred to as the Internet, coupled via a communications channel 125 to an internal network security system 110. The network system 100 further includes an internal computer network 115, such as a corporate Local Area Network (LAN), coupled via a communications channel 130 to the internal network computer system 110 and coupled via a communications channel 135 to a security management console 120.

The internal network security system 110 examines Downloadables received from external computer network 105, and prevents Downloadables deemed suspicious from reaching the internal computer network 115. It will be further appreciated that a Downloadable is deemed suspicious if it performs or may perform any undesirable operation, or if it threatens or may threaten the integrity of an internal computer network 115 component. It is to be understood that the term "suspicious" includes hostile, potentially hostile, undesirable, potentially undesirable, etc. Security management console 120 enables viewing, modification and configuration of the internal network security system 110.

FIG. 2 is a block diagram illustrating details of the internal network security system 110, which includes a Central Processing Unit (CPU) 205, such as an Intel Pentium® microprocessor or a Motorola Power PC® microprocessor, coupled to a signal bus 220. The internal network security system 110 further includes an external communications interface 210 coupled between the communications channel 125 and the signal bus 220 for receiving Downloadables from external computer network 105, and an internal communications interface 225 coupled between the signal bus 220 and the communications channel 130 for forwarding Downloadables not deemed suspicious to the internal computer network 115. The external communications interface 210 and the internal communications interface 225 may be functional components of an integral communications interface (not shown) for both receiving Downloadables from the external computer network 105 and forwarding Downloadables to the internal computer network 115.

Internal network security system 110 further includes Input/Output (I/O) interfaces 215 (such as a keyboard, mouse and Cathode Ray Tube (CRT) display), a data storage device 230 such as a magnetic disk, and a Random-Access Memory (RAM) 235, each coupled to the signal bus 220. The data storage device 230 stores a security database 240, which includes security information for determining whether a received Downloadable is to be deemed suspicious. The data storage device 230 further stores a users list 260 identifying the users within the internal computer network 115 who may receive Downloadables, and an event log 245 which includes determination results for each Downloadable examined and runtime indications of the internal network security system 110. An operating system 250 controls processing by CPU 205, and is typically stored in data storage device 230 and loaded into RAM 235 (as illustrated) for execution. A security program 255 controls examination of incoming Downloadables, and also may be stored in data storage device 230 and loaded into RAM 235 (as illustrated) for execution by CPU 205.

FIG. 3 is a block diagram illustrating details of the security program 255 and the security database 240. The security program 255 includes an ID generator 315, a policy finder 317 coupled to the ID generator 315, and a first comparator 320 coupled to the policy finder 317. The first comparator 320 is coupled to a logical engine 333 via four separate paths, namely, via Path 1, via Path 2, via Path 3 and via Path 4. Path 1 includes a direct connection from the first comparator 320 to the logical engine 333. Path 2 includes a code scanner coupled to the first comparator 320, and an Access Control List (ACL) comparator 330 coupling the code scanner 325 to the logical engine 333. Path 3 includes a certificate scanner 340 coupled to the first comparator 320, and a certificate comparator 345 coupling the certificate scanner 340 to the logical engine 333. Path 4 includes a Uniform Resource Locator (URL) comparator 350 coupling the first comparator 320 to the logical engine 3330. A record-keeping engine 335 is coupled between the logical engine 333 and the event log 245.

The security program 255 operates in conjunction with the security database 240, which includes security policies 305, known Downloadables 307, known Certificates 309 and Downloadable Security Profile (DSP) data 310 corresponding to the known Downloadables 307. Security policies 305 includes policies specific to particular users 260 and default (or generic) policies for determining whether to allow or block an incoming Downloadable. These security policies 305 may identify specific Downloadables to block, specific Downloadables to allow, or necessary criteria for allowing an unknown Downloadable. Referring to FIG. 4, security policies 305 include policy selectors 405, access control lists 410, trusted certificate lists 415, URL rule bases 420, and lists 425 of Downloadables to allow or to block per administrative override.

Known Downloadables 307 include lists of Downloadables which Original Equipment Manufacturers (OEMs) know to be hostile, of Downloadables which OEMs know to be non-hostile, and of Downloadables previously received by this security program 255. DSP data 310 includes the list of all potentially hostile or suspicious computer operations that may be attempted by each known Downloadable 307, and may also include the respective arguments of these operations. An identified argument of an operation is referred to as "resolved." An unidentified argument is referred to as "unresolved." DSP data 310 is described below with reference to the code scanner 325.

The ID generator 315 receives a Downloadable (including the URL from which it came and the userID of the intended recipient) from the external computer network 105 via the external communications interface 210, and generates a Downloadable ID for identifying each Downloadable. The Downloadable ID preferably includes a digital hash of the complete Downloadable code. The ID generator 315 preferably prefetches all components embodied in or identified by the code for Downloadable ID generation. For example, the ID generator 315 may prefetch all classes embodied in or identified by the Java™ applet bytecode to generate the Downloadable ID. Similarly, the ID generator 315 may retrieve all components listed in the INF file for an ActiveX™ control to compute a Downloadable ID. Accordingly, the Downloadable ID for the Downloadable will be the same each time the ID generator 315 receives the same Downloadable. The ID generator 315 adds the generated Downloadable ID to the list of known Downloadables

307 (if it is not already listed). The ID generator 315 then forwards the Downloadable and Downloadable ID to the policy finder 317.

The policy finder 317 uses the userID of the intended user and the Downloadable ID to select the specific security policy 305 that shall be applied on the received Downloadable. If there is a specific policy 305 that was defined for the user (or for one of its super groups) and the Downloadable, then the policy is selected. Otherwise the generic policy 305 that was defined for the user (or for one of its super groups) is selected. The policy finder 317 then sends the policy to the first comparator 320.

The first comparator 320 receives the Downloadable, the Downloadable ID and the security policy 305 from the policy finder 317. The first comparator 320 examines the security policy 305 to determine which steps are needed for allowing the Downloadable. For example, the security policy 305 may indicate that, in order to allow this Downloadable, it must pass all four paths, Path 1, Path 2, Path 3 and Path 4. Alternatively, the security policy 305 may indicate that to allow the Downloadable, the it must pass only one of the paths. The first comparator 320 responds by forwarding the proper information to the paths identified by the security policy 305.

Path 1

In path 1, the first comparator 320 checks the policy selector 405 of the security policy 305 that was received from the policy finder 317. If the policy selector 405 is either "Allowed" or "Blocked," then the first comparator 320 forwards this result directly to the logical engine 333. Otherwise, the first comparator 320 invokes the comparisons in path2 and/or path 3 and/or path 4 based on the contents of policy selector 405. It will be appreciated that the first comparator 320 itself compares the Downloadable ID against the lists of Downloadables to allow or block per administrative override 425. That is, the system security administrator can define specific Downloadables as "Allowed" or "Blocked."

Alternatively, the logical engine 333 may receive the results of each of the paths and based on the policy selector 405 may institute the final determination whether to allow or block the Downloadable. The first comparator 320 informs the logical engine 333 of the results of its comparison.

Path 2

In path 2, the first comparator 320 delivers the Downloadable, the Downloadable ID and the security policy 305 to the code scanner 325. If the DSP data 310 of the received Downloadable is known, the code scanner 325 retrieves and forwards the information to the ACL comparator 330. Otherwise, the code scanner 325 resolves the DSP data 310. That is, the code scanner 325 uses conventional parsing techniques to decompose the code (including all prefetched components) of the Downloadable into the DSP data 310. DSP data 310 includes the list of all potentially hostile or suspicious computer operations that may be attempted by a specific Downloadable 307, and may also include the respective arguments of these operations. For example, DSP data 310 may include a READ from a specific file, a SEND to an unresolved host, etc. The code scanner 325 may generate the DSP data 310 as a list of all operations in the Downloadable code which could ever be deemed potentially hostile and a list of all files to be accessed by the Downloadable code. It will be appreciated that the code scanner 325 may search the code for any pattern, which is undesirable or suggests that the code was written by a hacker.

An Example List of Operations Deemed Potentially Hostile

File operations: READ a file, WRITE a file;

Network operations: LISTEN on a socket, CONNECT to a socket, SEND data, RECEIVE data, VIEW INTRANET;

Registry operations: READ a registry item, WRITE a registry item;

Operating system operations: EXIT WINDOWS, EXIT BROWSER, START PROCESS/THREAD, KILL PROCESS/THREAD, CHANGE PROCESS/THREAD PRIORITY, DYNAMICALLY LOAD A CLASS/LIBRARY, etc.; and

Resource usage thresholds: memory, CPU, graphics, etc.

In the preferred embodiment, the code scanner 325 performs a full-content inspection. However, for improved speed but reduced security, the code scanner 325 may examine only a portion of the Downloadable such as the Downloadable header. The code scanner 325 then stores the DSP data into DSP data 310 (corresponding to its Downloadable ID), and sends the Downloadable, the DSP data to the ACL comparator 330 for comparison with the security policy 305.

The ACL comparator 330 receives the Downloadable, the corresponding DSP data and the security policy 305 from the code scanner 325, and compares the DSP data against the security policy 305. That is, the ACL comparator 330 compares the DSP data of the received Downloadable against the access control lists 410 in the received security policy 305. The access control list 410 contains criteria indicating whether to pass or fail the Downloadable. For example, an access control list may indicate that the Downloadable fails if the DSP data includes a WRITE command to a system file. The ACL comparator 330 sends its results to the logical engine 333.

Path 3

In path 3, the certificate scanner 340 determines whether the received Downloadable was signed by a certificate authority, such as VeriSign, Inc., and scans for a certificate embodied in the Downloadable. The certificate scanner 340 forwards the found certificate to the certificate comparator 345. The certificate comparator 345 retrieves known certificates 309 that were deemed trustworthy by the security administrator and compares the found certificate with the known certificates 309 to determine whether the Downloadable was signed by a trusted certificate. The certificate comparator 345 sends the results to the logical engine 333.

Path 4

In path 4, the URL comparator 350 examines the URL identifying the source of the Downloadable against URLs stored in the URL rule base 420 to determine whether the Downloadable comes from a trusted source. Based on the security policy 305, the URL comparator 350 may deem the Downloadable suspicious if the Downloadable comes from an untrustworthy source or if the Downloadable did not come from a trusted source. For example, if the Downloadable comes from a known hacker, then the Downloadable may be deemed suspicious and presumed hostile. The URL comparator 350 sends its results to the logical engine 333.

The logical engine 333 examines the results of each of the paths and the policy selector 405 in the security policy 305 to determine whether to allow or block the Downloadable. The policy selector 405 includes a logical expression of the results received from each of the paths. For example, the

logical engine 333 may block a Downloadable if it fails any one of the paths, i.e., if the Downloadable is known hostile (Path 1), if the Downloadable may request suspicious operations (Path 2), if the Downloadable was not signed by a trusted certificate authority (Path 3), or if the Downloadable did came from an untrustworthy source (Path 4). The logical engine 333 may apply other logical expressions according to the policy selector 405 embodied in the security policy 305. If the policy selector 405 indicates that the Downloadable may pass, then the logical engine 333 passes the Downloadable to its intended recipient. Otherwise, if the policy selector 405 indicates that the Downloadable should be blocked, then the logical engine 333 forwards a non-hostile Downloadable to the intended recipient to inform the user that internal network security system 110 discarded the original Downloadable. Further, the logical engine 333 forwards a status report to the record-keeping engine 335, which stores the reports in event log 245 in the data storage device 230 for subsequent review, for example, by the MIS director.

FIG. 5 is a block diagram illustrating details of the security management console 120, which includes a security policy editor 505 coupled to the communications channel 135, an event log analysis engine 510 coupled between communications channel 135 and a user notification engine 515, and a Downloadable database review engine 520 coupled to the communications channel 135. The security management console 120 further includes computer components similar to the computer components illustrated in FIG. 2.

The security policy editor 505 uses an I/O interface similar to I/O interface 215 for enabling authorized user modification of the security policies 305. That is, the security policy editor 505 enables the authorized user to modify specific security policies 305 corresponding to the users 260, the default or generic security policy 305, the Downloadables to block per administrative override, the Downloadables to allow per administrative override, the trusted certificate lists 415, the policy selectors 405, the access control lists 410, the URLs in the URL rule bases 420, etc. For example, if the authorized user learns of a new hostile Downloadable, then the user can add the Downloadable to the Downloadables to block per system override.

The event log analysis engine 510 examines the status reports contained in the event log 245 stored in the data storage device 230. The event log analysis engine 510 determines whether notification of the user (e.g., the security system manager or MIS director) is warranted. For example, the event log analysis engine 510 may warrant user notification whenever ten (10) suspicious Downloadables have been discarded by internal network security system 110 within a thirty (30) minute period, thereby flagging a potential imminent security threat. Accordingly, the event log analysis engine 510 instructs the user notification engine 515 to inform the user. The user notification engine 515 may send an e-mail via internal communications interface 220 or via external communications interface 210 to the user, or may display a message on the user's display device (not shown).

FIG. 6A is a flowchart illustrating a method 600 for protecting an internal computer network 115 from suspicious Downloadables. Method 600 begins with the ID generator 315 in step 602 receiving a Downloadable. The ID generator 315 in step 604 generates a Downloadable ID identifying the received Downloadable, preferably, by generating a digital hash of the Downloadable code (including prefetched components). The policy finder 317 in step 606 finds the appropriate security policy 305 corresponding to the userID specifying intended recipient (or the group to which the intended recipient belongs) and the Downloadable. The selected security policy 305 may be the default security policy 305. Step 606 is described in greater detail below with reference to FIG. 6B.

The first comparator 320 in step 608 examines the lists of Downloadables to allow or to block per administrative override 425 against the Downloadable ID of the incoming Downloadable to determine whether to allow the Downloadable automatically. If so, then in step 612 the first comparator 320 sends the results to the logical engine 333. If not, then the method 600 proceeds to step 610. In step 610, the first comparator 620 examines the lists of Downloadables to block per administrative override 425 against the Downloadable ID of the incoming Downloadable for determining whether to block the Downloadable automatically. If so, then the first comparator 420 in step 612 sends the results to the logical engine 333. Otherwise, method 600 proceeds to step 614.

In step 614, the first comparator 320 determines whether the security policy 305 indicates that the Downloadable should be tested according to Path 4. If not, then method 600 jumps to step 618. If so, then the URL comparator 350 in step 616 compares the URL embodied in the incoming Downloadable against the URLs of the URL rules bases 420, and then method 600 proceeds to step 618.

In step 618, the first comparator 320 determines whether the security policy 305 indicates that the Downloadable should be tested according to Path 2. If not, then method 600 jumps to step 620. Otherwise, the code scanner 235 in step 626 examines the DSP data 310 based on the Downloadable ID of the incoming Downloadable to determine whether the Downloadable has been previously decomposed. If so, then method 600 jumps to step 630. Otherwise, the code scanner 325 in step 628 decomposes the Downloadable into DSP data. Downloadable decomposition is described in greater detail with reference to FIG. 7. In step 630, the ACL comparator 330 compares the DSP data of the incoming Downloadable against the access control lists 410 (which include the criteria necessary for the Downloadable to fail or pass the test).

In step 620, the first comparator 320 determines whether the security policy 305 indicates that the Downloadable should be tested according to Path 3. If not, then method 600 returns to step 612 to send the results of each of the test performed to the logical engine 333. Otherwise, the certificate scanner 622 in step 622 scans the Downloadable for an embodied certificate. The certificate comparator 345 in step 624 retrieves trusted certificates from the trusted certificate lists (TCL) 415 and compares the embodied certificate with the trusted certificates to determine whether the Downloadable has been signed by a trusted source. Method 600 then proceeds to step 612 by the certificate scanner 345 sending the results of each of the paths taken to the logical engine 333. The operations of the logical engine 333 are described in greater detail below with reference to FIG. 6C. Method 600 then ends.

One skilled in the art will recognize that the tests may be performed in a different order, and that each of the tests need not be performed. Further, one skilled in the art will recognize that, although path 1 is described in FIG. 6A as an automatic allowance or blocking, the results of Path 1 may be another predicate to be applied by the logical engine 333. Further, although the tests are shown serially in FIG. 6A, the tests may be performed in parallel as illustrated in FIG. 3.

FIG. 6B is a flowchart illustrating details of step 606 of FIG. 6A (referred to herein as method 606). Method 606 begins with the policy finder 317 in step 650 determining whether security policies 305 include a specific security policy corresponding to the userID and the Downloadable. If so, then the policy finder 317 in step 654 fetches the corresponding specific policy 305. If not, then the policy finder 317 in step 652 fetches the default or generic security policy 305 corresponding to the userID. Method 606 then ends.

FIG. 6C is a flowchart illustrating details of a method 655 for determining whether to allow or to block the incoming Downloadable. Method 635 begins with the logical engine 333 in step 660 receiving the results from the first comparator 320, from the ACL comparator 330, from the certificate comparator 345 and from the URL comparator 350. The logical engine 333 in step 662 compares the results with the policy selector 405 embodied in the security policy 305, and in step 664 determines whether the policy selector 405 confirms the pass. For example, the policy selector 405 may indicate that the logical engine 333 pass the Downloadable if it passes one of the tests of Path 1, Path 2, Path 3 and Path 4. If the policy selector 405 indicates that the Downloadable should pass, then the logical engine 333 in step 666 passes the Downloadable to the intended recipient. In step 668, the logical engine 333 sends the results to the record-keeping engine 335, which in turn stores the results in the event log 245 for future review. Method 655 then ends. Otherwise, if the policy selector 405 in step 664 indicates that the Downloadable should not pass, then the logical engine 333 in step 670 stops the Downloadable and in step 672 sends a non-hostile substitute Downloadable to inform the user that the incoming Downloadable has been blocked. Method 655 then jumps to step 668.

FIG. 7 is a flowchart illustrating details of step 628 of FIG. 6A (referred to herein as method 628) for decomposing a Downloadable into DSP data 310. Method 628 begins in step 705 with the code scanner 325 disassembling the machine code of the Downloadable. The code scanner 325 in step 710 resolves a respective command in the machine code, and in step 715 determines whether the resolved command is suspicious (e.g., whether the command is one of the operations identified in the list described above with reference to FIG. 3). If not, then the code scanner 325 in step 725 determines whether it has completed decomposition of the Downloadable, i.e., whether all operations in the Downloadable code have been resolved. If so, then method 628 ends. Otherwise, method 628 returns to step 710.

Otherwise, if the code scanner 325 in step 71 determines that the resolved command is suspect, then the code scanner 325 in step 720 decodes and registers the suspicious command and its command parameters as DSP data 310. The code scanner 325 in step 720 registers the commands and command parameters into a format based on command class (e.g., file operations, network operations, registry operations, operating system operations, resource usage thresholds). Method 628 then jumps to step 725.

FIG. 8 is a flowchart illustrating a method 800 for generating a Downloadable ID for identifying a Downloadable. Method 800 begins with the ID generator 315 in step 810 receiving a Downloadable from the external computer network 105. The ID generator 315 in step 820 may fetch some or all components referenced in the Downloadable code, and in step 830 includes the fetched components in the Downloadable code. The ID generator 315 in step 840 performs a hashing function on at least a portion of the Downloadable code to generate a Downloadable ID. The ID generator 315 in step 850 stores the generated Downloadable ID in the security database 240 as a reference to the DSP data 310. Accordingly, the Downloadable ID will be the same for the identical Downloadable each time it is encountered.

The foregoing description of the preferred embodiments of the invention is by way of example only, and other variations of the above-described embodiments and methods are provided by the present invention. For example, although the invention has been described in a system for protecting an internal computer network, the invention can be embodied in a system for protecting an individual computer. Components of this invention may be implemented using a programmed general purpose digital computer, using application specific integrated circuits, or using a network of interconnected conventional components and circuits. The embodiments described herein have been presented for purposes of illustration and are not intended to be exhaustive or limiting. Many variations and modifications are possible in light of the foregoing teaching. The system is limited only by the following claims.

What is claimed is:

1. A computer-based method for generating a Downloadable ID to identify a Downloadable, comprising:

obtaining a Downloadable that includes one or more references to software components required to be executed by the Downloadable;

fetching at least one software component identified by the one or more references; and

performing a hashing function on the Downloadable and the fetched software components to generate a Downloadable ID.

2. The method of claim 1, wherein the Downloadable includes an applet.

3. The method of claim 1, wherein the Downloadable includes an active software control.

4. The method of claim 1, wherein the Downloadable includes a plugin.

5. The method of claim 1, wherein the Downloadable includes HTML code.

6. The method of claim 1, wherein the Downloadable includes an application program.

7. The method of claim 1, wherein said fetching includes fetching a first software component referenced by the Downloadable.

8. The method of claim 1, wherein said fetching includes fetching all software components referenced by the Downloadable.

9. A system for generating a Downloadable ID to identify a Downloadable, comprising:

a communications engine for obtaining a Downloadable that includes one or more references to software components required to be executed by the Downloadable; and

an ID generator coupled to the communications engine that fetches at least one software component identified by the one or more references, and for performing a hashing function on the Downloadable and the fetched software components to generate a Downloadable ID.

10. The system of claim 9, wherein the Downloadable includes an applet.

11. The system of claim 9, wherein the Downloadable includes an active software control.

12. The system of claim 9, wherein the Downloadable includes a plugin.

13. The system of claim 9, wherein the Downloadable includes HTML code.

14. The system of claim 9, wherein the Downloadable includes an application program.

15. The system of claim 9, wherein the ID generator fetches a first software component referenced by the Downloadable.

16. The method of claim 9, wherein the ID generator fetches all software components referenced by the Downloadable.

17. A system for generating a Downloadable ID to identify a Downloadable, comprising:

means for obtaining a Downloadable that includes one or more references to software components required to be executed by the Downloadable;

means for fetching at least one software component identified by the one or more references; and

means for performing a hashing function on the Downloadable and the fetched software components to generate a Downloadable ID.

18. A computer-readable storage medium storing program code for causing a computer to perform the steps of:

obtaining a Downloadable that includes one or more references to software components required to be executed by the Downloadable;

fetching at least one software component identified by the one or more references; and

performing a hashing function on the Downloadable and the fetched software components to generate a Downloadable ID.

* * * * *

US007058822B2

(12) **United States Patent**
Edery et al.

(10) **Patent No.: US 7,058,822 B2**
(45) **Date of Patent: Jun. 6, 2006**

(54) **MALICIOUS MOBILE CODE RUNTIME MONITORING SYSTEM AND METHODS**

(75) Inventors: **Yigal Mordechai Edery**, Pardesia (IL); **Nimrod Itzhak Vered**, Geosh Tel-Mond (IL); **David R. Kroll**, San Jose, CA (US)

(73) Assignee: **Finjan Software, Ltd.**, South Netanya (IL)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1013 days.

(21) Appl. No.: **09/861,229**

(22) Filed: **May 17, 2001**

(65) **Prior Publication Data**

US 2002/0013910 A1 Jan. 31, 2002

Related U.S. Application Data

(63) Continuation-in-part of application No. 09/551,302, filed on Apr. 18, 2000, now Pat. No. 6,480,962, which is a continuation-in-part of application No. 09/539,667, filed on Mar. 30, 2000, now Pat. No. 6,804,780.

(60) Provisional application No. 60/205,591, filed on May 17, 2000.

(51) Int. Cl.
G06F 11/30 (2006.01)

(52) U.S. Cl. 713/200

(58) Field of Classification Search 713/176, 713/175, 200, 201, 150, 168; 701/223, 229; 717/120, 124, 126, 127, 130, 131, 134, 135; 709/223–229
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,077,677 A 12/1991 Murphy et al.
5,359,659 A 10/1994 Rosenthal
5,361,359 A 11/1994 Tajalli et al.
5,485,409 A 1/1996 Gupta et al.

(Continued)

OTHER PUBLICATIONS

Zhong et al, "Security in the large: is Java's sandbox scalable?", Oct. 1998, Seventh IEEE Symposium on Reliable Distributed Systems, pp 1-6.*

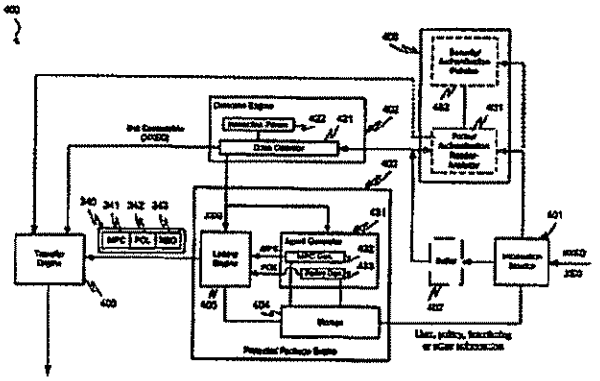
(Continued)

Primary Examiner—Christopher Revak
(74) *Attorney, Agent, or Firm*—Squire, Sanders & Dempsey, L.L.P.

(57) **ABSTRACT**

Protection systems and methods provide for protecting one or more personal computers ("PCs") and/or other intermittently or persistently network accessible devices or processes from undesirable or otherwise malicious operations of Java™ applets, ActiveX™ controls, JavaScript™ scripts, Visual Basic scripts, add-ins, downloaded/uploaded programs or other "Downloadables" or "mobile code" in whole or part. A protection engine embodiment provides, within a server, firewall or other suitable "re-communicator," for monitoring information received by the communicator, determining whether received information does or is likely to include executable code, and if so, causes mobile protection code (MPC) to be transferred to and rendered operable within a destination device of the received information, more suitably by forming a protection agent including the MPC, protection policies and a detected-Downloadable. An MPC embodiment further provides, within a Downloadable-destination, for initiating the Downloadable, enabling malicious Downloadable operation attempts to be received by the MPC, and causing (predetermined) corresponding operations to be executed in response to the attempts, more suitably in conjunction with protection policies.

35 Claims, 10 Drawing Sheets



U.S. PATENT DOCUMENTS

5,485,575	A	1/1996	Chess et al.	
5,572,643	A	11/1996	Judson	
5,606,668	A	2/1997	Shwed	
5,623,600	A	4/1997	Ji et al.	
5,638,446	A	6/1997	Rubin	
5,692,047	A	11/1997	McManis	
5,692,124	A	11/1997	Holden et al.	
5,720,033	A	2/1998	Deo	
5,724,425	A	3/1998	Chang et al.	
5,740,248	A	4/1998	Fieres et al.	
5,761,421	A	6/1998	van Hoff et al.	
5,765,205	A	6/1998	Breslau et al.	
5,784,459	A	7/1998	Devarakonda et al.	
5,796,952	A	8/1998	Davis et al.	
5,805,829	A	9/1998	Cohen et al.	
5,832,208	A	11/1998	Chen et al.	
5,850,559	A	12/1998	Angelo et al.	
5,859,966	A	1/1999	Hayman et al.	
5,864,683	A	1/1999	Boebert et al.	
5,892,904	A	4/1999	Atkinson et al.	
5,951,698	A	9/1999	Chen et al.	
5,956,481	A	9/1999	Walsh et al.	
5,974,549	A	10/1999	Golan	
5,978,484	A	11/1999	Apperson et al.	
5,983,348	A	11/1999	Ji	
6,092,194	A	7/2000	Touboul	
6,154,844	A	11/2000	Touboul et al.	
6,167,520	A	12/2000	Touboul	
6,425,058	B1	7/2002	Arimilli et al.	
6,434,668	B1	8/2002	Arimilli et al.	
6,434,669	B1	8/2002	Arimilli et al.	
6,480,962	B1	11/2002	Touboul	
6,519,679	B1	2/2003	Devireddy et al.	
6,732,179	B1 *	5/2004	Brown et al.	709/229

OTHER PUBLICATIONS

Rubin et al, "Mobile code security" Dec. 1998, IEEE Internet, pp 30-34.*

Schmid et al, "Protecting data from malicious software", 2002, Proceeding of the 18th Annual Computer Security Applications Conference, pp 1-10.*

Corradi et al, "A flexible access control service for Java mobile code", 2000, IEEE, pp 356-365.*

Jim K. Omura, "Novel Applications of Cryptography in Digital Communications", IEEE Communications Magazine, May, 1990; pp. 21-29.

Okamoto, E. et al., "ID-Based Authentication System For Computer Virus Detection", IEEE/IEE Electronic Library online, Electronics Letters, vol. 26, Issue 15, ISSN 0013-5194, Jul. 19, 1990, Abstract and pp. 1169-1170. URL:http://iel.ihs.com:80/cgi-bin/iel_cgi?se...2ehts%26ViewTemplate%3ddocview%5fb%2ehts.

IBM AntiVirus User's Guide Version 2.4, International Business Machines Corporation, Nov. 15, 1995, p. 6-7.

Norvin Leach et al, "IE 3.0 Applets Will Earn Certification", PC Week vol. 13, No. 29, Jul. 22, 1996, 2 pages.

"Finjan Software Releases SurfinBoard, Industry's First JAVA Security Product For the World Wide Web", Article published on the Internet by Finjan Software Ltd., Jul. 29, 1996, 1 page.

"Powerful PC Security for the New World of Java™ and Downloadables, Surfin Shield™" Article published on the Internet by Finjan Software Ltd., 1996, 2 Pages.

Microsoft® Authenticode Technology, "Ensuring Accountability and Authenticity for Software Components on the Internet", Microsoft Corporation, Oct. 1996, including Abstract, Contents, Introduction and pp. 1-10.

"Finjan Announces a Personal Java™ Firewall For Web Browsers—the SurfinShield™ 1.6 (formerly known as SurfinBoard)", Press Release of Finjan Releases SurfinShield 1.6, Oct. 21, 1996, 2 pages.

Company Profile "Finjan—Safe Surfing, The Java Security Solutions Provider" Article published on the Internet by Finjan Software Ltd., Oct. 31, 1996, 3 pages.

"Finjan Announces Major Power Boost and New Features for SurfinShield™ 2.0" Las Vegas Convention Center/Pavilion 5 P5551, Nov. 18, 1996, 3 pages.

"Java Security: Issues & Solutions" Articles published on the Internet by Finjan Software Ltd., 1996, 8 pages.

"Products" Articles published on the Internet, 7 pages.

Mark LaDue, "Online Business Consultant: Java Security: Whose Business Is It?" Article published on the Internet, Home Page Press, Inc. 1996, 4 pages.

Ron Moritz, "Why We Shouldn't Fear Java." Java Report, Feb., 1997, pp. 51-56.

Web Page Article "Frequently Asked Questions About Authenticode", Microsoft Corporation, last updated Feb. 17, 1997, Printed Dec. 23, 1998. URL: http://www.microsoft.com/workshop/security/authcode/signfaq.asp#9, pp. 1-13.

Zhang, X.N., "Secure Code Distribution", IEEE/IEE Electronic Library online, Computer, vol. 30, Issue 6, Jun., 1997, pp.: 76-79.

Khare, Rohit, "Microsoft Authenticode Analyzed", Jul. 22, 1996, 2 pages. URL: http://www.xent.com/FoRK-archive/summer96/0338.html.

"Release Notes for the Microsoft ActiveX Development Kit", Aug. 13, 1996, 11 pages URL: http://activex.adsp.or.jp/inetsdk/readme.txt.

"Microsoft ActiveX Software Development Kit", Aug. 12, 1996, 6 pages. URL: http://activex.adsp.or.jp/inetsdk/help.overview.htm.

* cited by examiner

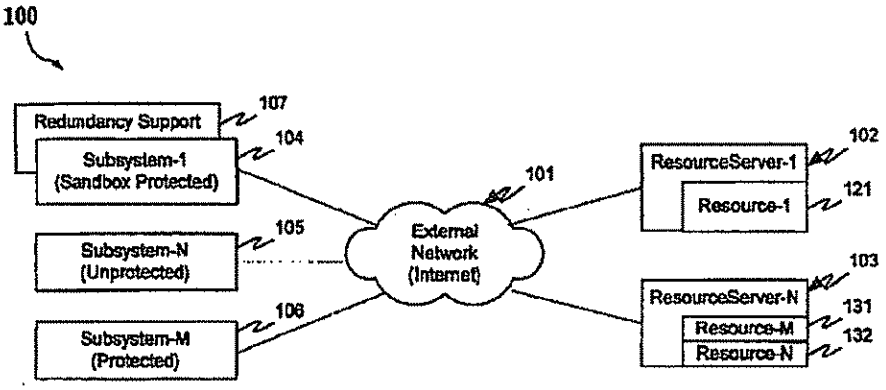


FIG. 1a

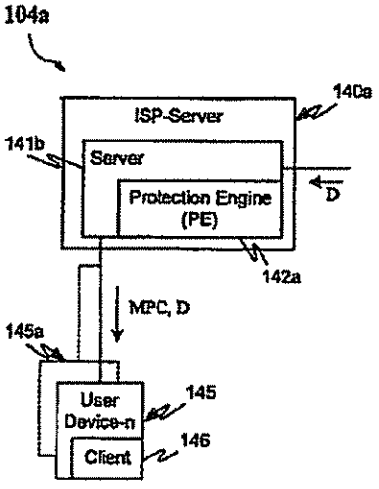


FIG. 1b

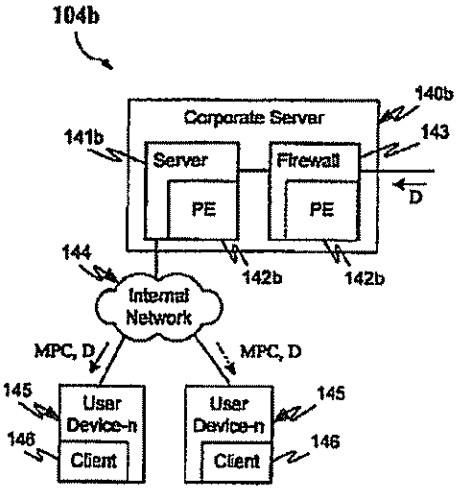


FIG. 1c

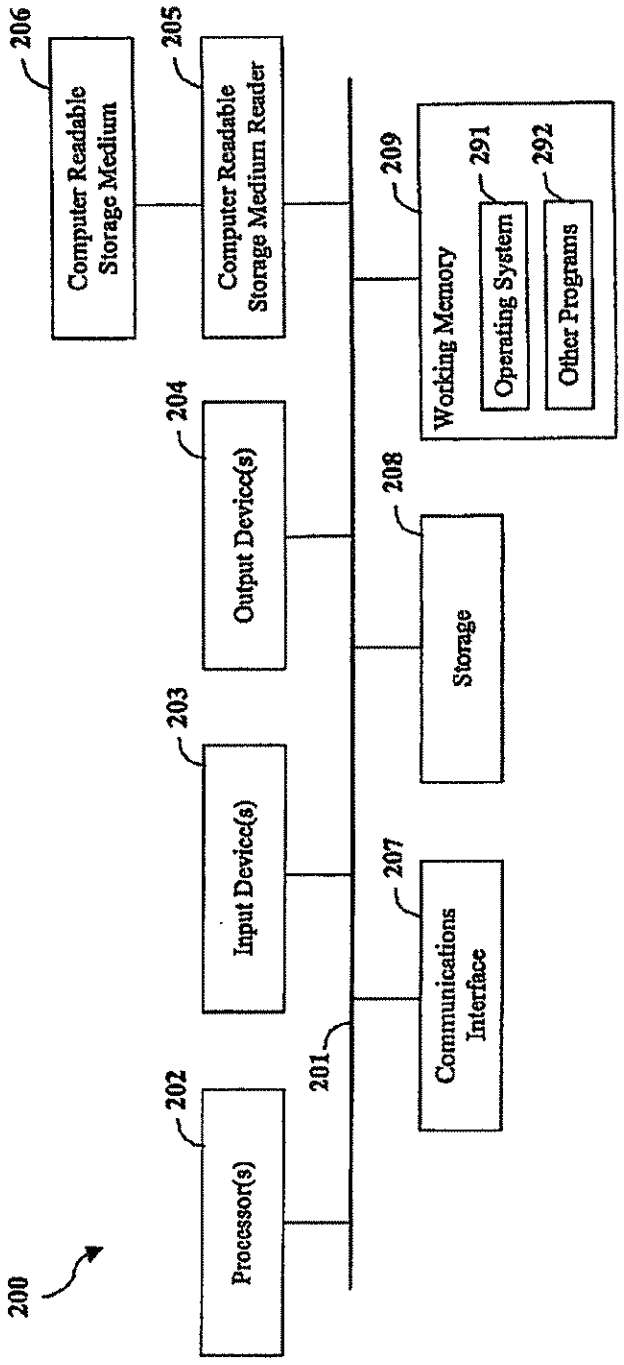

200
Processor(s)
202
201
Input Device(s)
203
Output Device(s)
204
Computer Readable Storage Medium Reader
205
Computer Readable Storage Medium
206
Communications Interface
207
Storage
208
Working Memory
209
Operating System
291
Other Programs
292

FIG. 2

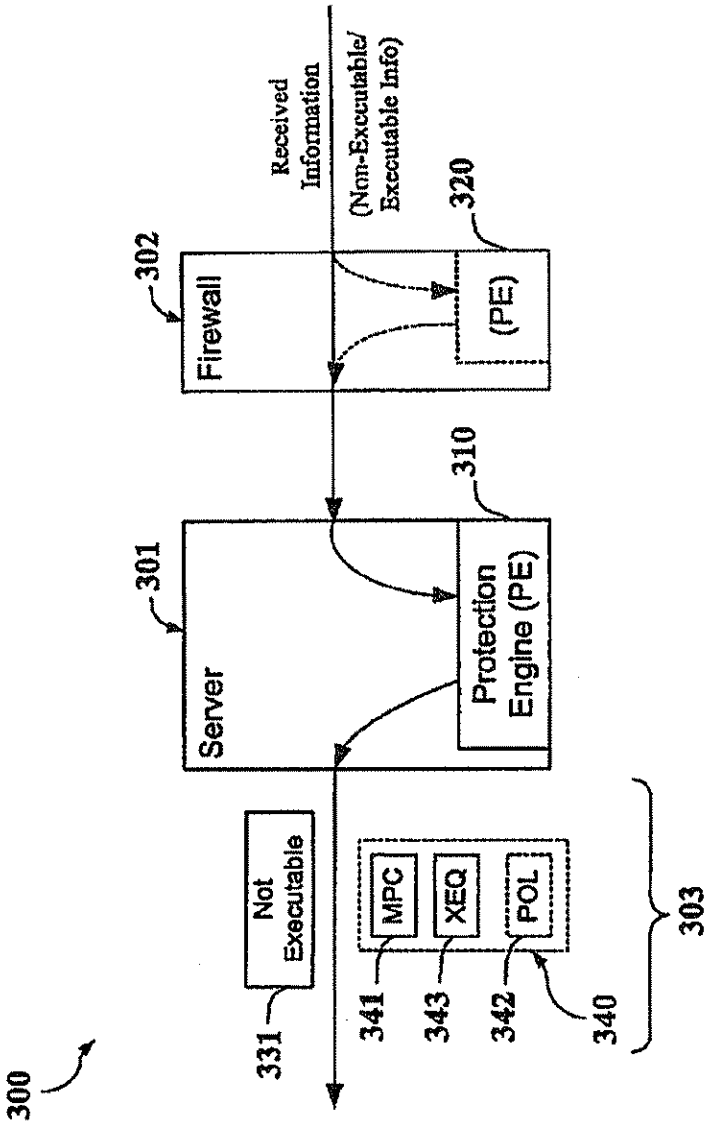

Received Information
(Non-Executable/ Executable Info)
320
302
Firewall
(PE)
310
301
Server
Protection Engine (PE)
Not Executable
MPC
XEQ
POL
303
300
331
341
343
342
340

FIG. 3

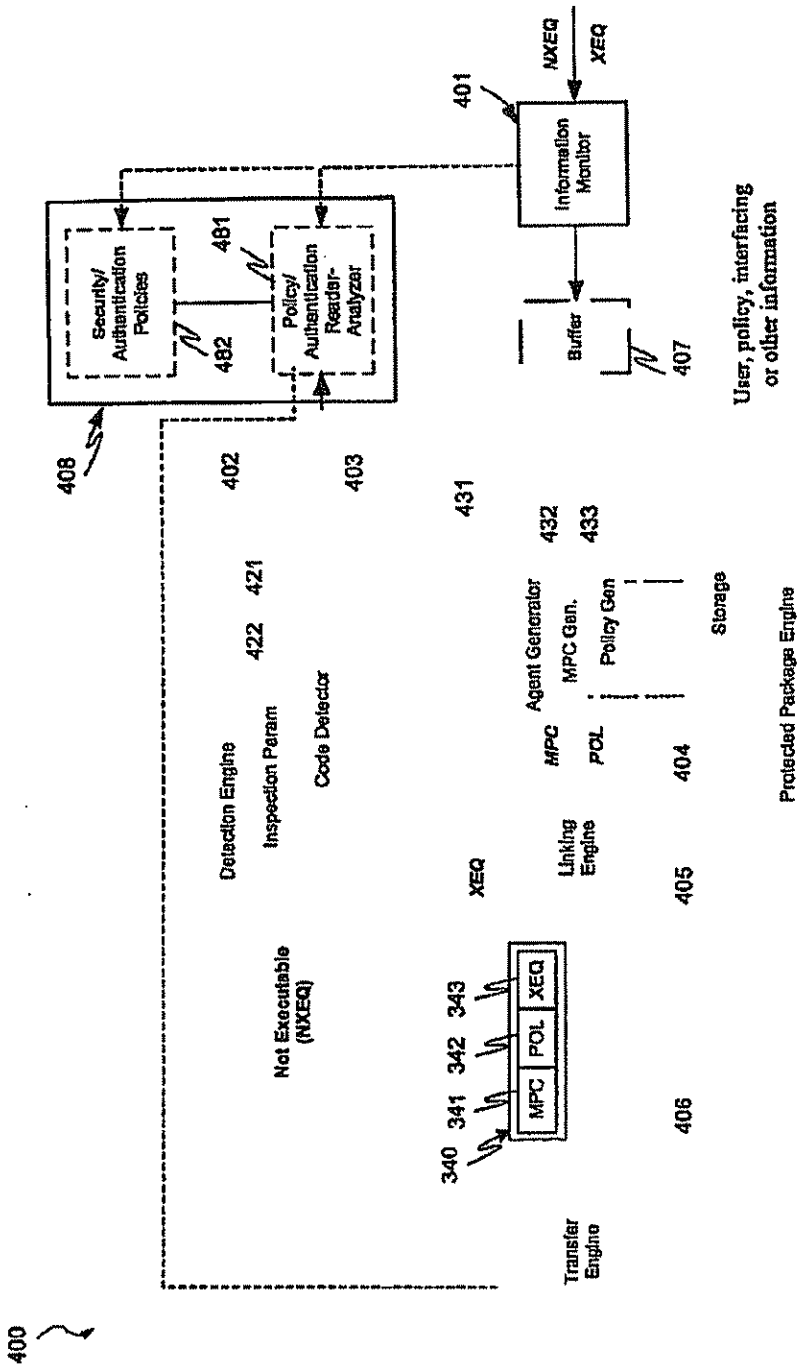

400
401
NXEQ
XEQ
Information Monitor
User, policy, interfacing or other information
Buffer
407
481
Security/ Authentication Policies
Policy/ Authentication Reader-Analyzer
482
408
402
403
431
432
433
421
422
Agent Generator
MPC Gen.
Policy Gen
Storage
Protected Package Engine
Detection Engine
Inspection Param
Code Detector
MPC
POL
404
Linking Engine
XEQ
405
Not Executable (NXEQ)
343
342
341
340
MPC
POL
XEQ
406
Transfer Engine

FIG. 4

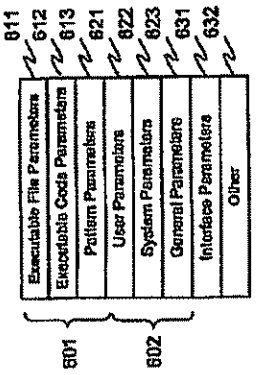


FIG. 6a

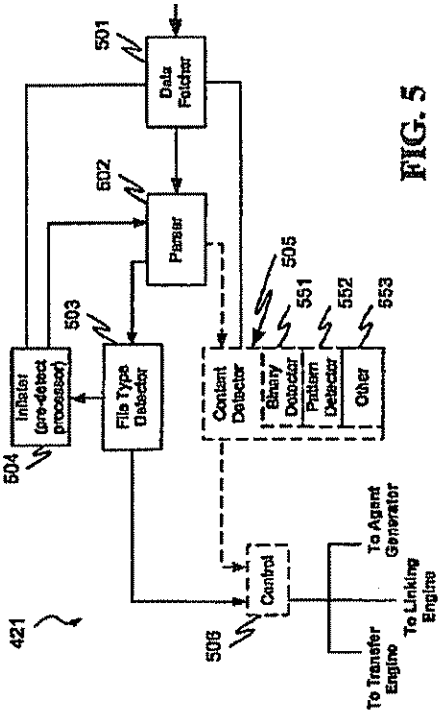


FIG. 5

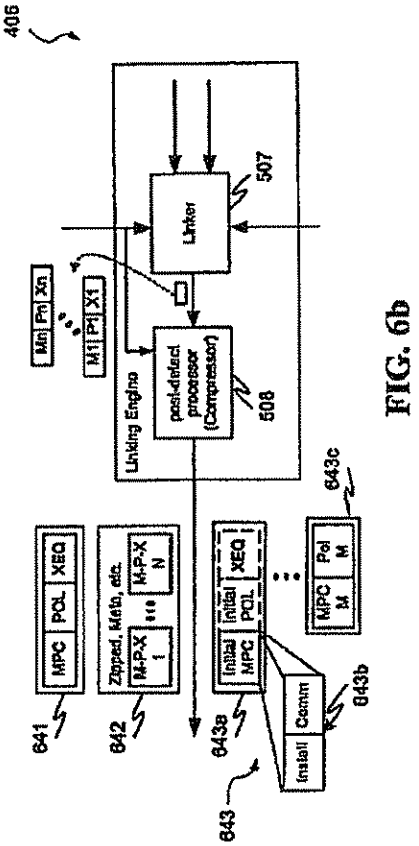


FIG. 6b

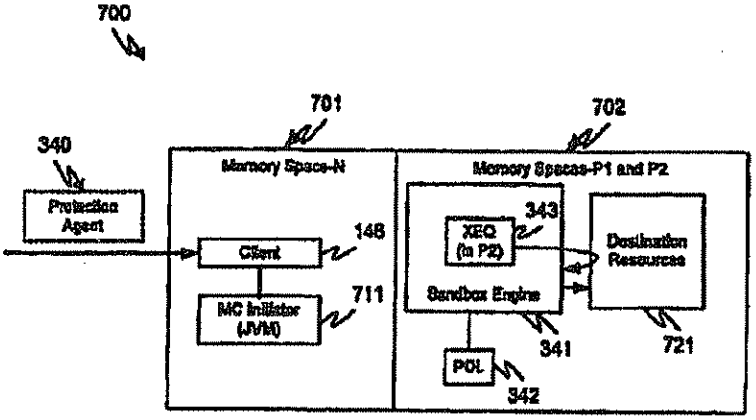


FIG. 7a

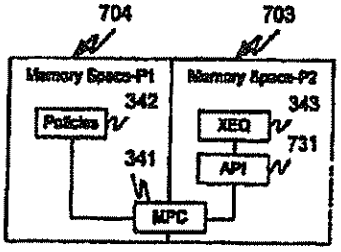


FIG. 7b

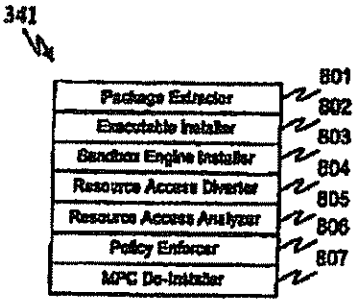


FIG. 8

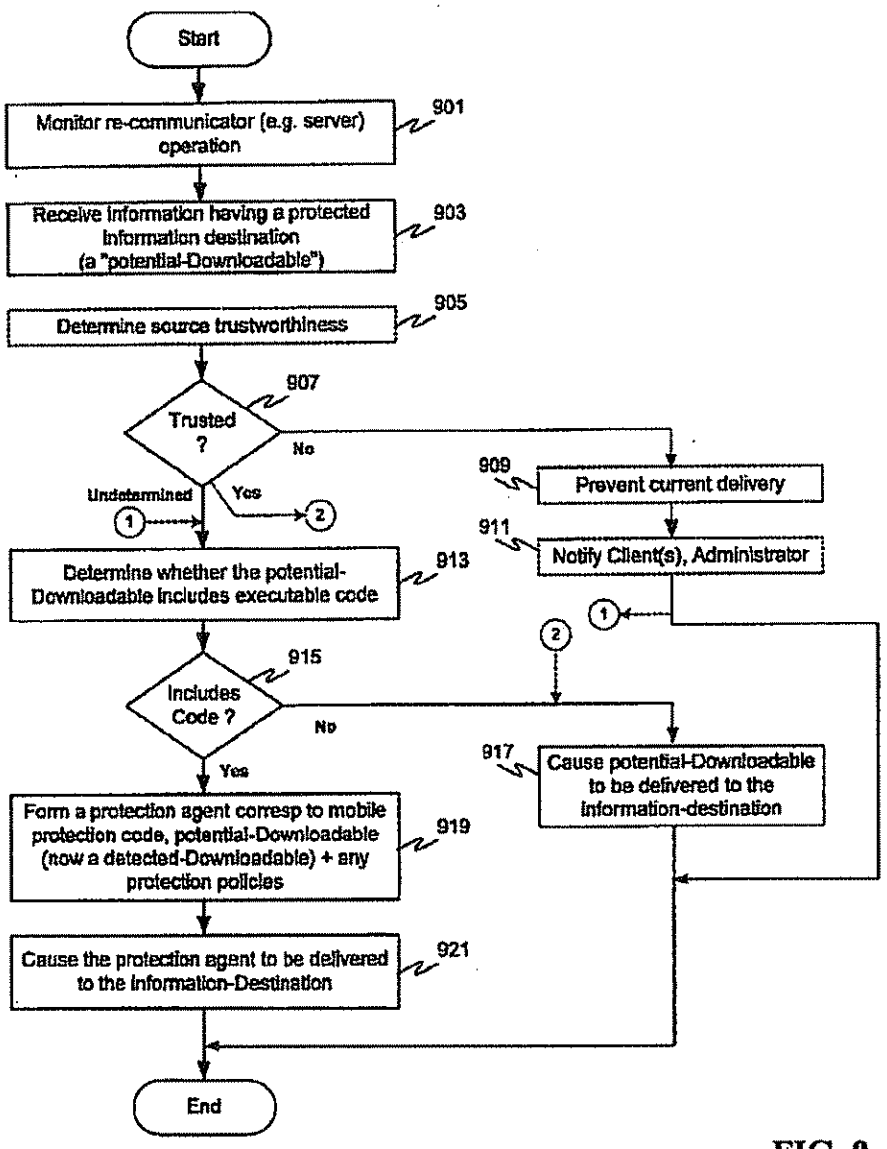

Start
Monitor re-communicator (e.g. server) operation
901
Receive information having a protected information destination (a "potential-Downloadable")
903
Determine source trustworthiness
905
Trusted ?
907
No
Undetermined
Yes
1
2
909
Prevent current delivery
911
Notify Client(s), Administrator
Determine whether the potential-Downloadable includes executable code
913
Includes Code ?
915
No
Yes
917
Cause potential-Downloadable to be delivered to the information-destination
Form a protection agent corresp to mobile protection code, potential-Downloadable (now a detected-Downloadable) + any protection policies
919
Cause the protection agent to be delivered to the information-Destination
921
End

FIG. 9

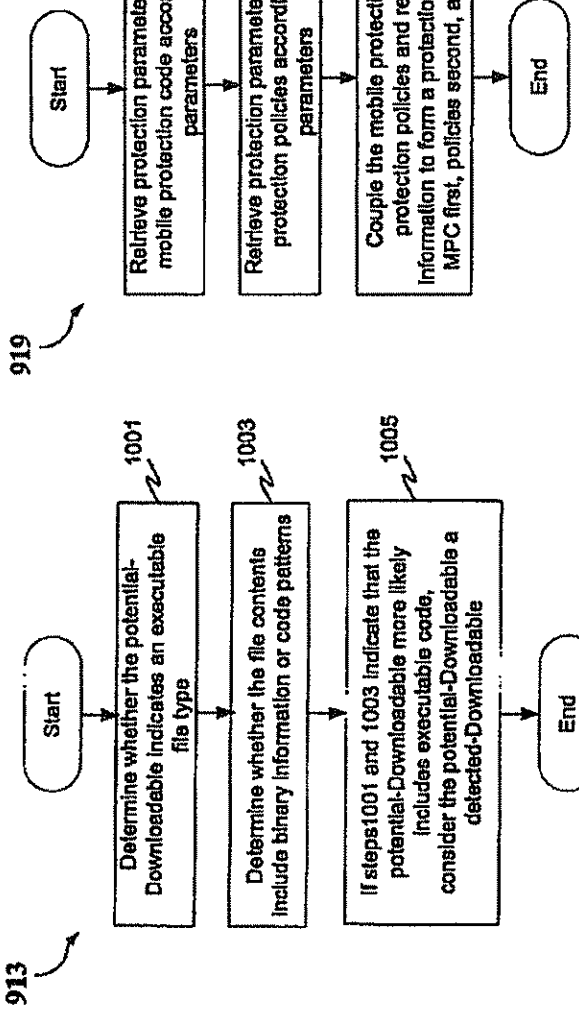


FIG. 10A

FIG. 10B

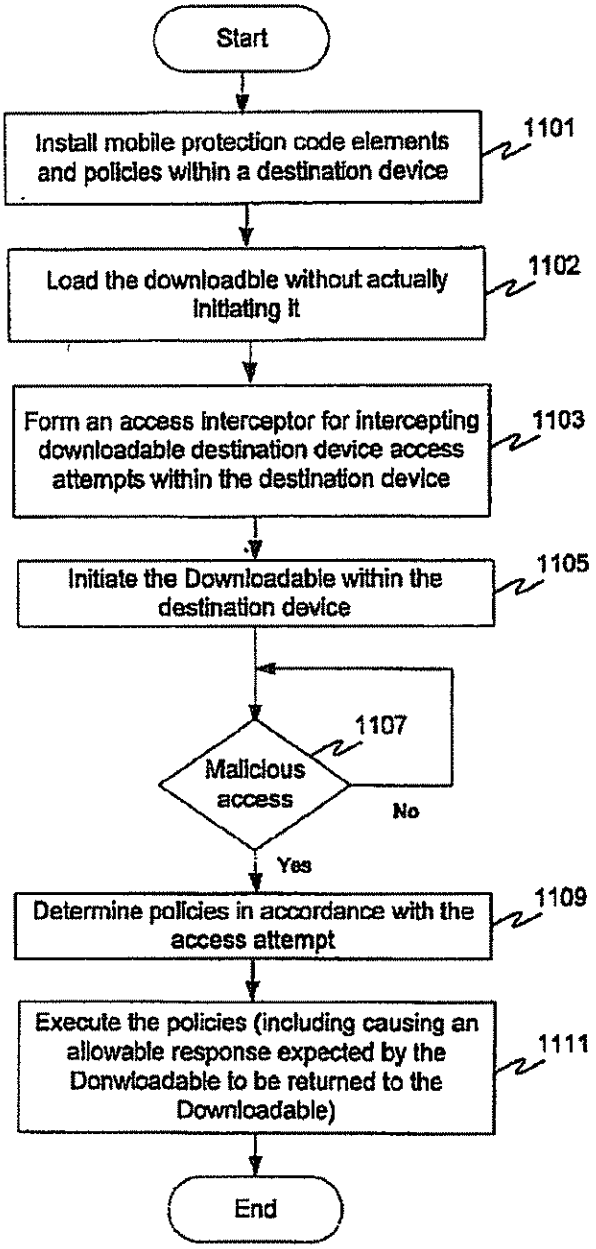

Start
Install mobile protection code elements and policies within a destination device
1101
Load the downloadble without actually initiating it
1102
Form an access interceptor for intercepting downloadable destination device access attempts within the destination device
1103
Initiate the Downloadable within the destination device
1105
Malicious access
1107
No
Yes
Determine policies in accordance with the access attempt
1109
Execute the policies (including causing an allowable response expected by the Donwloadable to be returned to the Downloadable)
1111
End

FIG. 11

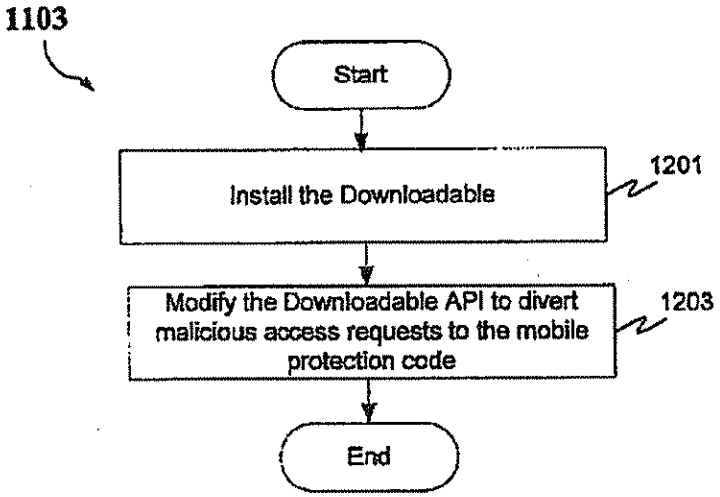


FIG. 12a

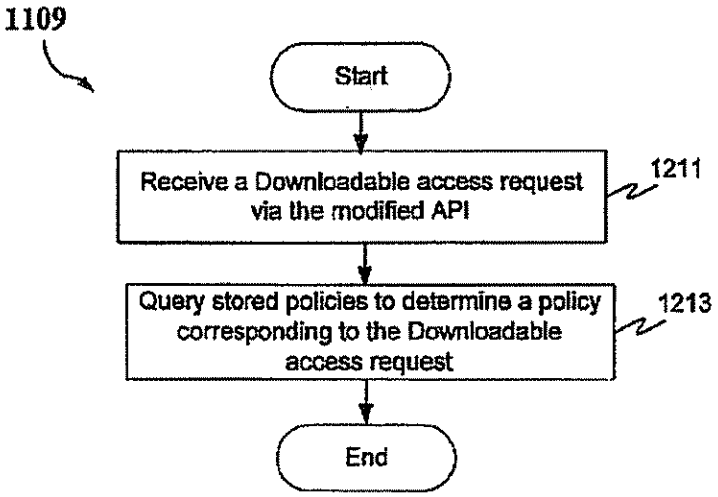


FIG. 12b

MALICIOUS MOBILE CODE RUNTIME MONITORING SYSTEM AND METHODS

PRIORITY REFERENCE TO RELATED APPLICATIONS

This application claims benefit of and hereby incorporates by reference provisional application Ser. No. 60/205,591, entitled "Computer Network Malicious Code Run-time Monitoring," filed on May 17, 2000 by inventors Nimrod Itzhak Vered, et al. This application is also a Continuation-In-Part of and hereby incorporates by reference patent application Ser. No. 09/539,667, now U.S. Pat. No. 6,804,780, entitled "System and Method for Protecting a Computer and a Network From Hostile Downloadables" filed on Mar. 30, 2000 by inventor Shlomo Touboul. This application is also a Continuation-In-Part of and hereby incorporates by reference patent application Ser. No. 09/551,302, now U.S. Pat. No. 6,480,962, entitled "System and Method for Protecting a Client During Runtime From Hostile Downloadables", filed on Apr. 18, 2000 by inventor Shlomo Touboul.

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates generally to computer networks, and more particularly provides a system and methods for protecting network-connectable devices from undesirable downloadable operation.

2. Description of the Background Art

Advances in networking technology continue to impact an increasing number and diversity of users. The Internet, for example, already provides to expert, intermediate and even novice users the informational, product and service resources of over 100,000 interconnected networks owned by governments, universities, nonprofit groups, companies, etc. Unfortunately, particularly the Internet and other public networks have also become a major source of potentially system-fatal or otherwise damaging computer code commonly referred to as "viruses."

Efforts to forestall viruses from attacking networked computers have thus far met with only limited success at best. Typically, a virus protection program designed to identify and remove or protect against the initiating of known viruses is installed on a network firewall or individually networked computer. The program is then inevitably surmounted by some new virus that often causes damage to one or more computers. The damage is then assessed and, if isolated, the new virus is analyzed. A corresponding new virus protection program (or update thereof) is then developed and installed to combat the new virus, and the new program operates successfully until yet another new virus appears—and so on. Of course, damage has already typically been incurred.

To make matters worse, certain classes of viruses are not well recognized or understood, let alone protected against. It is observed by this inventor, for example, that Downloadable information comprising program code can include distributable components (e.g. Java™ applets and JavaScript scripts, ActiveX™ controls, Visual Basic, add-ins and/or others). It can also include, for example, application programs, Trojan horses, multiple compressed programs such as zip or meta files, among others. U.S. Pat. No. 5,983,348 to Shuang, however, teaches a protection system for protecting against only distributable components including "Java applets or ActiveX controls", and further does so using resource intensive and high bandwidth static Downloadable

content and operational analysis, and modification of the Downloadable component; Shuang further fails to detect or protect against additional program code included within a tested Downloadable. U.S. Pat. No. 5,974,549 to Golan teaches a protection system that further focuses only on protecting against ActiveX controls and not other distributable components, let alone other Downloadable types. U.S. Pat. No. 6,167,520 to Touboul enables more accurate protection than Shuang or Golan, but lacks the greater flexibility and efficiency taught herein, as do Shuang and Golan.

Accordingly, there remains a need for efficient, accurate and flexible protection of computers and other network connectable devices from malicious Downloadables.

SUMMARY OF THE INVENTION

The present invention provides protection systems and methods capable of protecting a personal computer ("PC") or other persistently or even intermittently network accessible devices or processes from harmful, undesirable, suspicious or other "malicious" operations that might otherwise be effectuated by remotely operable code. While enabling the capabilities of prior systems, the present invention is not nearly so limited, resource intensive or inflexible, and yet enables more reliable protection. For example, remotely operable code that is protectable against can include downloadable application programs, Trojan horses and program code groupings, as well as software "components", such as Java™ applets, ActiveX™ controls, JavaScript™/Visual Basic scripts, add-ins, etc., among others. Protection can also be provided in a distributed interactively, automatically or mixed configurable manner using protected client, server or other parameters, redirection, local/remote logging, etc., and other server/client based protection measures can also be separately and/or interoperably utilized, among other examples.

In one aspect, embodiments of the invention provide for determining, within one or more network "servers" (e.g. firewalls, resources, gateways, email relays or other devices/processes that are capable of receiving-and-transferring a Downloadable) whether received information includes executable code (and is a "Downloadable"). Embodiments also provide for delivering static, configurable and/or extensible remotely operable protection policies to a Downloadable-destination, more typically as a sandboxed package including the mobile protection code, downloadable policies and one or more received Downloadables. Further client-based or remote protection code/policies can also be utilized in a distributed manner. Embodiments also provide for causing the mobile protection code to be executed within a Downloadable-destination in a manner that enables various Downloadable operations to be detected, intercepted or further responded to via protection operations. Additional server/information-destination device security or other protection is also enabled, among still further aspects.

A protection engine according to an embodiment of the invention is operable within one or more network servers, firewalls or other network connectable information re-communicating devices (as are referred to herein summarily one or more "servers" or "re-communicators"). The protection engine includes an information monitor for monitoring information received by the server, and a code detection engine for determining whether the received information includes executable code. The protection engine also includes a packaging engine for causing a sandboxed package, typically including mobile protection code and downloadable protection policies to be sent to a Downloadable-

destination in conjunction with the received information, if the received information is determined to be a Downloadable.

A sandboxed package according to an embodiment of the invention is receivable by and operable with a remote Downloadable-destination. The sandboxed package includes mobile protection code ("MPC") for causing one or more predetermined malicious operations or operation combinations of a Downloadable to be monitored or otherwise intercepted. The sandboxed package also includes protection policies (operable alone or in conjunction with further Downloadable-destination stored or received policies/MPCs) for causing one or more predetermined operations to be performed if one or more undesirable operations of the Downloadable is/are intercepted. The sandboxed package can also include a corresponding Downloadable and can provide for initiating the Downloadable in a protective "sandbox". The MPC/policies can further include a communicator for enabling further MPC/policy information or "modules" to be utilized and/or for event logging or other purposes.

A sandbox protection system according to an embodiment of the invention comprises an installer for enabling a received MPC to be executed within a Downloadable-destination (device/process) and further causing a Downloadable application program, distributable component or other received downloadable code to be received and installed within the Downloadable-destination. The protection system also includes a diverter for monitoring one or more operation attempts of the Downloadable, an operation analyzer for determining one or more responses to the attempts, and a security enforcer for effectuating responses to the monitored operations. The protection system can further include one or more security policies according to which one or more protection system elements are operable automatically (e.g. programmatically) or in conjunction with user intervention (e.g. as enabled by the security enforcer). The security policies can also be configurable/extensible in accordance with further downloadable and/or Downloadable-destination information.

A method according to an embodiment of the invention includes receiving downloadable information, determining whether the downloadable information includes executable code, and causing a mobile protection code and security policies to be communicated to a network client in conjunction with security policies and the downloadable information if the downloadable information is determined to include executable code. The determining can further provide multiple tests for detecting, alone or together, whether the downloadable information includes executable code.

A further method according to an embodiment of the invention includes forming a sandboxed package that includes mobile protection code ("MPC"), protection policies, and a received, detected-Downloadable, and causing the sandboxed package to be communicated to and installed by a receiving device or process ("user device") for responding to one or more malicious operation attempts by the detected-Downloadable from within the user device. The MPC/policies can further include a base "module" and a "communicator" for enabling further up/downloading of one or more further "modules" or other information (e.g. events, user/user device information, etc.).

Another method according to an embodiment of the invention includes installing, within a user device, received mobile protection code ("MPC") and protection policies in conjunction with the user device receiving a downloadable application program, component or other Downloadable(s). The method also includes determining, by the MPC, a resource access attempt by the Downloadable, and initiating, by the MPC, one or more predetermined operations corresponding to the attempt. (Predetermined operations can, for example, comprise initiating user, administrator, client, network or protection system determinable operations, including but not limited to modifying the Downloadable operation, extricating the Downloadable, notifying a user/another, maintaining a local/remote log, causing one or more MPCs/policies to be downloaded, etc.)

Advantageously, systems and methods according to embodiments of the invention enable potentially damaging, undesirable or otherwise malicious operations by even unknown mobile code to be detected, prevented, modified and/or otherwise protected against without modifying the mobile code. Such protection is further enabled in a manner that is capable of minimizing server and client resource requirements, does not require pre-installation of security code within a Downloadable-destination, and provides for client specific or generic and readily updatable security measures to be flexibly and efficiently implemented. Embodiments further provide for thwarting efforts to bypass security measures (e.g. by "hiding" undesirable operation causing information within apparently inert or otherwise "friendly" downloadable information) and/or dividing or combining security measures for even greater flexibility and/or efficiency.

Embodiments also provide for determining protection policies that can be downloaded and/or ascertained from other security information (e.g. browser settings, administrative policies, user input, uploaded information, etc.). Different actions in response to different Downloadable operations, clients, users and/or other criteria are also enabled, and embodiments provide for implementing other security measures, such as verifying a downloadable source, certification, authentication, etc. Appropriate action can also be accomplished automatically (e.g. programmatically) and/or in conjunction with alerting one or more users/administrators, utilizing user input, etc. Embodiments further enable desirable Downloadable operations to remain substantially unaffected, among other aspects.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1*a* is a block diagram illustrating a network system in accordance with an embodiment of the present invention;

FIG. 1*b* is a block diagram illustrating a network subsystem example in accordance with an embodiment of the invention;

FIG. 1*c* is a block diagram illustrating a further network subsystem example in accordance with an embodiment of the invention;

FIG. 2 is a block diagram illustrating a computer system in accordance with an embodiment of the invention;

FIG. 3 is a flow diagram broadly illustrating a protection system host according to an embodiment of the invention;

FIG. 4 is a block diagram illustrating a protection engine according to an embodiment of the invention;

FIG. 5 is a block diagram illustrating a content inspection engine according to an embodiment of the invention;

FIG. 6*a* is a block diagram illustrating protection engine parameters according to an embodiment of the invention;

FIG. 6*b* is a flow diagram illustrating a linking engine use in conjunction with ordinary, compressed and distributable sandbox package utilization, according to an embodiment of the invention;

FIG. 7*a* is a flow diagram illustrating a sandbox protection system operating within a destination system, according to an embodiment of the invention;

FIG. 7*b* is a block diagram illustrating memory allocation usable in conjunction with the protection system of FIG. 7*a*, according to an embodiment of the invention;

FIG. 8 is a block diagram illustrating a mobile protection code according to an embodiment of the invention;

FIG. 9 is a flowchart illustrating a server based protection method according to an embodiment of the invention;

FIG. 10*a* is a flowchart illustrating method for determining if a potential-Downloadable includes or is likely to include executable code, according to an embodiment of the invention;

FIG. 10*b* is a flowchart illustrating a method for forming a protection agent, according to an embodiment of the invention;

FIG. 11 is a flowchart illustrating a method for protecting a Downloadable destination according to an embodiment of the invention;

FIG. 12*a* is a flowchart illustrating a method for forming a Downloadable access interceptor according to an embodiment of the invention; and

FIG. 12*b* is a flowchart illustrating a method for implementing mobile protection policies according to an embodiment of the invention.

DETAILED DESCRIPTION

In providing malicious mobile code runtime monitoring systems and methods, embodiments of the invention enable actually or potentially undesirable operations of even unknown malicious code to be efficiently and flexibly avoided. Embodiments provide, within one or more "servers" (e.g. firewalls, resources, gateways, email relays or other information re-communicating devices), for receiving downloadable-information and detecting whether the downloadable-information includes one or more instances of executable code (e.g. as with a Trojan horse, zip/meta file etc.). Embodiments also provide for separately or interoperably conducting additional security measures within the server, within a Downloadable-destination of a detected-Downloadable, or both.

Embodiments further provide for causing mobile protection code ("MPC") and downloadable protection policies to be communicated to, installed and executed within one or more received information destinations in conjunction with a detected-Downloadable. Embodiments also provide, within an information-destination, for detecting malicious operations of the detected-Downloadable and causing responses thereto in accordance with the protection policies (which can correspond to one or more user, Downloadable, source, destination, or other parameters), or further downloaded or downloadable-destination based policies (which can also be configurable or extensible). (Note that the term "or", as used herein, is generally intended to mean "and/or" unless otherwise indicated.)

FIGS. 1*a* through 1*c* illustrate a computer network system 100 according to an embodiment of the invention. FIG. 1*a* broadly illustrates system 100, while FIGS. 1*b* and 1*c* illustrate exemplary protectable subsystem implementations corresponding with system 104 or 106 of FIG. 1*a*.

Beginning with FIG. 1*a*, computer network system 100 includes an external computer network 101, such as a Wide Area Network or "WAN" (e.g. the Internet), which is coupled to one or more network resource servers (summarily depicted as resource server-1 102 and resource server-N 103). Where external network 101 includes the Internet, resource servers 1-N (102, 103) might provide one or more resources including web pages, streaming media, transaction-facilitating information, program updates or other downloadable information, summarily depicted as resources 121, 131 and 132. Such information can also include more traditionally viewed "Downloadables" or "mobile code" (i.e. distributable components), as well as downloadable application programs or other further Downloadables, such as those that are discussed herein. (It will be appreciated that interconnected networks can also provide various other resources as well.)

Also coupled via external network 101 are subsystems 104–106. Subsystems 104–106 can, for example, include one or more servers, personal computers ("PCs"), smart appliances, personal information managers or other devices/processes that are at least temporarily or otherwise intermittently directly or indirectly connectable in a wired or wireless manner to external network 101 (e.g. using a dialup, DSL, cable modem, cellular connection, IR/RF, or various other suitable current or future connection alternatives). One or more of subsystems 104–106 might further operate as user devices that are connectable to external network 101 via an internet service provider ("ISP") or local area network ("LAN"), such as a corporate intranet, or home, portable device or smart appliance network, among other examples.

FIG. 1*a* also broadly illustrates how embodiments of the invention are capable of selectively, modifiably or extensibly providing protection to one or more determinable ones of networked subsystems 104–106 or elements thereof (not shown) against potentially harmful or other undesirable ("malicious") effects in conjunction with receiving downloadable information. "Protected" subsystem 104, for example, utilizes a protection in accordance with the teachings herein, while "unprotected" subsystem-N 105 employs no protection, and protected subsystem-M 106 might employ one or more protections including those according to the teachings herein, other protection, or some combination.

System 100 implementations are also capable of providing protection to redundant elements 107 of one or more of subsystems 104–106 that might be utilized, such as backups, failsafe elements, redundant networks, etc. Where included, such redundant elements are also similarly protectable in a separate, combined or coordinated manner using embodiments of the present invention either alone or in conjunction with other protection mechanisms. In such cases, protection can be similarly provided singly, as a composite of component operations or in a backup fashion. Care should, however, be exercised to avoid potential repeated protection engine execution corresponding to a single Downloadable; such "chaining" can cause a Downloadable to operate incorrectly or not at all, unless a subsequent detection engine is configured to recognize a prior packaging of the Downloadable.

FIGS. 1*b* and 1*c* further illustrate, by way of example, how protection systems according to embodiments of the invention can be utilized in conjunction with a wide variety of different system implementations. In the illustrated examples, system elements are generally configurable in a manner commonly referred to as a "client-server" configuration, as is typically utilized for accessing Internet and many other network resources. For clarity sake, a simple client-server configuration will be presumed unless otherwise indicated. It will be appreciated, however, that other configurations of interconnected elements might also be utilized (e.g. peer-peer, routers, proxy servers, networks,

converters, gateways, services, network reconfiguring elements, etc.) in accordance with a particular application.

The FIG. 1*b* example shows how a suitable protected system 104*a* (which can correspond to subsystem-1 104 or subsystem-M 106 of FIG. 1) can include a protection-initiating host "server" or "re-communicator" (e.g. ISP server 140*a*), one or more user devices or "Downloadable-destinations" 145, and zero or more redundant elements (which elements are summarily depicted as redundant client device/process 145*a*). In this example, ISP server 140*a* includes one or more email, Internet or other servers 141*a*, or other devices or processes capable of transferring or otherwise "re-communicating" downloadable information to user devices 145. Server 141*a* further includes protection engine or "PE" 142*a*, which is capable of supplying mobile protection code ("MPC") and protection policies for execution by client devices 145. One or more of user devices 145 can further include a respective one or more clients 146 for utilizing information received via server 140*a*, in accordance with which MPC and protection policies are operable to protect user devices 145 from detrimental, undesirable or otherwise "malicious" operations of downloadable information also received by user device 145.

The FIG. 1*c* example shows how a further suitable protected system 104*b* can include, in addition to a "re-communicator", such as server 142*b*, a firewall 143*c* (e.g. as is typically the case with a corporate intranet and many existing or proposed home/smart networks.) In such cases, a server 141*b* or firewall 143 can operate as a suitable protection engine host. A protection engine can also be implemented in a more distributed manner among two or more protection engine host systems or host system elements, such as both of server 141*b* and firewall 143, or in a more integrated manner, for example, as a standalone device. Redundant system or system protection elements can also be similarly provided in a more distributed or integrated manner (see above).

System 104*b* also includes internal network 144 and user devices 145. User devices 145 further include a respective one or more clients 146 for utilizing information received via server 140*a*, in accordance with which the MPCs or protection policies are operable. (As in the previous example, one or more of user devices 145 can also include or correspond with similarly protectable redundant system elements, which are not shown.)

It will be appreciated that the configurations of FIGS. 1*a*–1*c* are merely exemplary. Alternative embodiments might, for example, utilize other suitable connections, devices or processes. One or more devices can also be configurable to operate as a network server, firewall, smart router, a resource server servicing deliverable third-party/manufacturer postings, a user device operating as a firewall/server, or other information-suppliers or intermediaries (i.e. as a "re-communicator" or "server") for servicing one or more further interconnected devices or processes or interconnected levels of devices or processes. Thus, for example, a suitable protection engine host can include one or more devices or processes capable of providing or supporting the providing of mobile protection code or other protection consistent with the teachings herein. A suitable information-destination or "user device" can further include one or more devices or processes (such as email, browser or other clients) that are capable of receiving and initiating or otherwise hosting a mobile code execution.

FIG. 2 illustrates an exemplary computing system 200, that can comprise one or more of the elements of FIGS. 1*a* through 1*c*. While other application-specific alternatives might be utilized, it will be presumed for clarity sake that system 100 elements (FIGS. 1*a–c*) are implemented in hardware, software or some combination by one or more processing systems consistent therewith, unless otherwise indicated.

Computer system 200 comprises elements coupled via communication channels (e.g. bus 201) including one or more general or special purpose processors 202, such as a Pentium® or Power PC®, digital signal processor ("DSP"), etc. System 200 elements also include one or more input devices 203 (such as a mouse, keyboard, microphone, pen, etc.), and one or more output devices 204, such as a suitable display, speakers, actuators, etc., in accordance with a particular application.

System 200 also includes a computer readable storage media reader 205 coupled to a computer readable storage medium 206, such as a storage/memory device or hard or removable storage/memory media; such devices or media are further indicated separately as storage device 208 and memory 209, which can include hard disk variants, floppy/compact disk variants, digital versatile disk ("DVD") variants, smart cards, read only memory, random access memory, cache memory, etc., in accordance with a particular application. One or more suitable communication devices 207 can also be included, such as a modem, DSL, infrared or other suitable transceiver, etc. for providing inter-device communication directly or via one or more suitable private or public networks that can include but are not limited to those already discussed.

Working memory further includes operating system ("OS") elements and other programs, such as application programs, mobile code, data, etc. for implementing system 100 elements that might be stored or loaded therein during use. The particular OS can vary in accordance with a particular device, features or other aspects in accordance with a particular application (e.g. Windows, Mac, Linux, Unix or Palm OS variants, a proprietary OS, etc.). Various programming languages or other tools can also be utilized, such as C++, Java, Visual Basic, etc. As will be discussed, embodiments can also include a network client such as a browser or email client, e.g. as produced by Netscape, Microsoft or others, a mobile code executor such as an OS task manager, Java Virtual Machine ("JVM"), etc., and an application program interface ("API"), such as a Microsoft Windows or other suitable element in accordance with the teachings herein. (It will also become apparent that embodiments might also be implemented in conjunction with a resident application or combination of mobile code and resident application components.)

One or more system 200 elements can also be implemented in hardware, software or a suitable combination. When implemented in software (e.g. as an application program, object, downloadable, servlet, etc. in whole or part), a system 200 element can be communicated transitionally or more persistently from local or remote storage to memory (or cache memory, etc.) for execution, or another suitable mechanism can be utilized, and elements can be implemented in compiled or interpretive form. Input, intermediate or resulting data or functional elements can further reside more transitionally or more persistently in a storage media, cache or more persistent volatile or non-volatile memory, (e.g. storage device 207 or memory 208) in accordance with a particular application.

FIG. 3 illustrates an interconnected re-communicator 300 generally consistent with system 140*b* of FIG. 1, according to an embodiment of the invention. As with system 140*b*, system 300 includes a server 301, and can also include a

firewall 302. In this implementation, however, either server 301 or firewall 302 (if a firewall is used) can further include a protection engine (310 or 320 respectively). Thus, for example, an included firewall can process received information in a conventional manner, the results of which can be further processed by protection engine 310 of server 301, or information processed by protection engine 320 of an included firewall 302 can be processed in a conventional manner by server 301. (For clarity sake, a server including a singular protection engine will be presumed, with or without a firewall, for the remainder of the discussion unless otherwise indicated. Note, however, that other embodiments consistent with the teachings herein might also be utilized.)

FIG. 3 also shows how information received by server 301 (or firewall 302) can include non-executable information, executable information or a combination of non-executable and one or more executable code portions (e.g. so-called Trojan horses that include a hostile Downloadable within a friendly one, combined, compressed or otherwise encoded files, etc.). Particularly such combinations will likely remain undetected by a firewall or other more conventional protection systems. Thus, for convenience, received information will also be referred to as a "potential-Downloadable", and received information found to include executable code will be referred to as a "Downloadable" or equivalently as a "detected-Downloadable" (regardless of whether the executable code includes one or more application programs, distributable "components" such as Java, ActiveX, add-in, etc.).

Protection engine 310 provides for detecting whether received potential-Downloadables include executable code, and upon such detection, for causing mobile protection code ("MPC") to be transferred to a device that is a destination of the potential-Downloadable (or "Downloadable-destination"). Protection engine 310 can also provide protection policies in conjunction with the MPC (or thereafter as well), which MPC/policies can be automatically (e.g. programmatically) or interactively configurable in accordance user, administrator, downloadable source, destination, operation, type or various other parameters alone or in combination (see below). Protection engine 310 can also provide or operate separately or interoperably in conjunction with one or more of certification, authentication, downloadable tagging, source checking, verification, logging, diverting or other protection services via the MPC, policies, other local/remote server or destination processing, etc. (e.g. which can also include protection mechanisms taught by the above-noted prior applications; see FIG. 4).

Operationally, protection engine 310 of server 301 monitors information received by server 301 and determines whether the received information is deliverable to a protected destination, e.g. using a suitable monitor/data transfer mechanism and comparing a destination-address of the received information to a protected destination set, such as a protected destinations list, array, database, etc. (All deliverable information or one or more subsets thereof might also be monitored.) Protection engine 310 further analyzes the potential-Downloadable and determines whether the potential-Downloadable includes executable code. If not, protection engine 310 enables the not executable potential-Downloadable 331 to be delivered to its destination in an unaffected manner.

In conjunction with determining that the potential-Downloadable is a detected-Downloadable, protection engine 310 also causes mobile protection code or "MPC" 341 to be communicated to the Downloadable-destination of the Downloadable, more suitably in conjunction with the detected-Downloadable 343 (see below). Protection engine 310 further causes downloadable protection policies 342 to be delivered to the Downloadable-destination, again more suitably in conjunction with the detected-Downloadable. Protection policies 342 provide parameters (or can additionally or alternatively provide additional mobile code) according to which the MPC is capable of determining or providing applicable protection to a Downloadable-destination against malicious Downloadable operations.

(One or more "checked", tag, source, destination, type, detection or other security result indicators, which are not shown, can also be provided as corresponding to determined non-Downloadables or Downloadables, e.g. for testing, logging, further processing, further identification tagging or other purposes in accordance with a particular application.)

Further MPCs, protection policies or other information are also deliverable to a the same or another destination, for example, in accordance with communication by an MPC/protection policies already delivered to a downloadable-destination. Initial or subsequent MPCs/policies can further be selected or configured in accordance with a Downloadable-destination indicated by the detected-Downloadable, destination-user or administrative information, or other information providable to protection engine 310 by a user, administrator, user system, user system examination by a communicated MPC, etc. (Thus, for example, an initial MPC/policies can also be initially provided that are operable with or optimized for more efficient operation with different Downloadable-destinations or destination capabilities.)

While integrated protection constraints within the MPC might also be utilized, providing separate protection policies has been found to be more efficient, for example, by enabling more specific protection constraints to be more easily updated in conjunction with detected-Downloadable specifics, post-download improvements, testing, etc. Separate policies can further be more efficiently provided (e.g. selected, modified, instantiated, etc.) with or separately from an MPC, or in accordance with the requirements of a particular user, device, system, administration, later improvement, etc., as might also be provided to protection engine 310 (e.g. via user/MPC uploading, querying, parsing a Downloadable, or other suitable mechanism implemented by one or more servers or Downloadable-destinations).

(It will also become apparent that performing executable code detection and communicating to a downloadable-Destination an MPC and any applicable policies as separate from a detected-Downloadable is more accurate and far less resource intensive than, for example, performing content and operation scanning, modifying a Downloadable, or providing completely Downloadable-destination based security.)

System 300 enables a single or extensible base-MPC to be provided, in anticipation or upon receipt of a first Downloadable, that is utilized thereafter to provide protection of one or more Downloadable-destinations. It is found, however, that providing an MPC upon each detection of a Downloadable (which is also enabled) can provide a desirable combination of configurability of the MPC/policies and lessened need for management (e.g. given potentially changing user/destination needs, enabling testing, etc.).

Providing an MPC upon each detection of a Downloadable also facilitates a lessened demand on destination resources, e.g. since information-destination resources used in executing the MPC/policies can be re-allocated following such use. Such alternatives can also be selectively, modifiably or extensibly provided (or further in accordance with other application-specific factors that might also apply.)

Thus, for example, a base-MPC or base-policies might be provided to a user device that is/are extensible via additionally downloadable "modules" upon server 301 detection of a Downloadable deliverable to the same user device, among other alternatives.

In accordance with a further aspect of the invention, it is found that improved efficiency can also be achieved by causing the MPC to be executed within a Downloadable-destination in conjunction with, and further, prior to initiation of the detected Downloadable. One mechanism that provides for greater compatibility and efficiency in conjunction with conventional client-based Downloadable execution is for a protection engine to form a sandboxed package 340 including MPC 341, the detected-Downloadable 343 and any policies 342. For example, where the Downloadable is a binary executable to be executed by an operating system, protection engine 310 forms a protected package by concatenating, within sandboxed package 340, MPC 341 for delivery to a Downloadable-destination first, followed by protection policies 342 and Downloadable 343. (Concatenation or techniques consistent therewith can also be utilized for providing a protecting package corresponding to a Java applet for execution by a JVM of a Downloadable-destination, or with regard to ActiveX controls, add-ins or other distributable components, etc.)

The above concatenation or other suitable processing will result in the following. Upon receipt of sandboxed package 340 by a compatible browser, email or other destination-client and activating of the package by a user or the destination-client, the operating system (or a suitable responsively initiated distributed component host) will attempt to initiate sandboxed package 340 as a single Downloadable. Such processing will, however, result in initiating the MPC 341 and—in accordance with further aspects of the invention—the MPC will initiate the Downloadable in a protected manner, further in accordance with any applicable included or further downloaded protection policies 342. (While system 300 is also capable of ascertaining protection policies stored at a Downloadable-destination, e.g. by poll, query, etc. of available destination information, including at least initial policies within a suitable protecting package is found to avoid associated security concerns or inefficiencies.)

Turning to FIG. 4, a protection engine 400 generally consistent with protection engine 310 (or 320) of FIG. 3 is illustrated in accordance with an embodiment of the invention. Protection engine 400 comprises information monitor 401, detection engine 402, and protected packaging engine 403, which further includes agent generator 431, storage 404, linking engine 405, and transfer engine 406. Protection engine 400 can also include a buffer 407, for temporarily storing a received potential-Downloadable, or one or more systems for conducting additional authentication, certification, verification or other security processing (e.g. summarily depicted as security system 408) Protection engine 400 can further provide for selectively re-directing, further directing, logging, etc. of a potential/detected Downloadable or information corresponding thereto in conjunction with detection, other security, etc., in accordance with a particular application.

(Note that FIG. 4, as with other figures included herein, also depicts exemplary signal flow arrows; such arrows are provided to facilitate discussion, and should not be construed as exclusive or otherwise limiting.)

Information monitor 401 monitors potential-Downloadables received by a host server and provides the information via buffer 407 to detection engine 402 or to other system 400 elements. Information monitor 401 can be configured to monitor host server download operations in conjunction with a user or a user-device that has logged-on to the server, or to receive information via a server operation hook, servlet, communication channel or other suitable mechanism.

Information monitor 401 can also provide for transferring, to storage 404 or other protection engine elements, configuration information including, for example, user, MPC, protection policy, interfacing or other configuration information (e.g. see FIG. 6). Such configuration information monitoring can be conducted in accordance with a user/device logging onto or otherwise accessing a host server, via one or more of configuration operations, using an applet to acquire such information from or for a particular user, device or devices, via MPC/policy polling of a user device, or via other suitable mechanisms.

Detection engine 402 includes code detector 421, which receives a potential-Downloadable and determines, more suitably in conjunction with inspection parameters 422, whether the potential-Downloadable includes executable code and is thus a "detected-Downloadable". (Code detector 421 can also include detection processors for performing file decompression or other "decoding", or such detection-facilitating processing as decryption, utilization/support of security system 408, etc. in accordance with a particular application.)

Detection engine 402 further transfers a detected-downloadable ("XEQ") to protected packaging engine 403 along with indicators of such detection, or a determined non-executable ("NXEQ") to transfer engine 406. (Inspection parameters 422 enable analysis criteria to be readily updated or varied, for example, in accordance with particular source, destination or other potential Downloadable impacting parameters, and are discussed in greater detail with reference to FIG. 5). Detection engine 402 can also provide indicators for delivery of initial and further MPCs/policies, for example, prior to or in conjunction with detecting a Downloadable and further upon receipt of an indicator from an already downloaded MPC/policy. A downloaded MPC/policy can further remain resident at a user device with further modules downloaded upon or even after delivery of a sandboxed package. Such distribution can also be provided in a configurable manner, such that delivery of a complete package or partial packages are automatically or interactively determinable in accordance with user/administrative preferences/policies, among other examples.

Packaging engine 403 provides for generating mobile protection code and protection policies, and for causing delivery thereof (typically with a detected-Downloadable) to a Downloadable-destination for protecting the Downloadable-destination against malicious operation attempts by the detected Downloadable. In this example, packaging engine 403 includes agent generator 431, storage 404 and linking engine 405.

Agent generator 431 includes an MPC generator 432 and a protection policy generator 433 for "generating" an MPC and a protection policy (or set of policies) respectively upon receiving one or more "generate MPC/policy" indicators from detection engine 402, indicating that a potential-Downloadable is a detected-Downloadable. MPC generator 432 and protection policy generator 433 provide for generating MPCs and protection policies respectively in accordance with parameters retrieved from storage 404. Agent generator 431 is further capable of providing multiple MPCs/policies, for example, the same or different MPCs/policies in accordance with protecting ones of multiple executables within a

zip file, or for providing initial MPCs/policies and then further MPCs/policies or MPC/policy "modules" as initiated by further indicators such as given above, via an indicator of an already downloaded MPC/policy or via other suitable mechanisms. (It will be appreciated that pre-constructed MPCs/policies or other processing can also be utilized, e.g. via retrieval from storage 404, but with a potential decrease in flexibility.)

MPC generator 432 and protection policy generator 433 are further configurable. Thus, for example, more generic MPCs/policies can be provided to all or a grouping of serviced destination-devices (e.g. in accordance with a similarly configured/administered intranet), or different MPCs/policies that can be configured in accordance with one or more of user, network administration, Downloadable-destination or other parameters (e.g. see FIG. 6). As will become apparent, a resulting MPC provides an operational interface to a destination device/process. Thus, a high degree of flexibility and efficiency is enabled in providing such an operational interface within different or differently configurable user devices/processes or other constraints.

Such configurability further enables particular policies to be utilized in accordance with a particular application (e.g. particular system uses, access limitations, user interaction, treating application programs or Java components from a particular known source one way and unknown source ActiveX components, or other considerations). Agent generator 431 further transfers a resulting MPC and protection policy pair to linking engine 405.

Linking engine 405 provides for forming from received component elements (see above) a sandboxed package that can include one or more initial or complete MPCs and applicable protection policies, and a Downloadable, such that the sandboxed package will protect a receiving Downloadable-destination from malicious operation by the Downloadable. Linking engine 405 is implementable in a static or configurable manner in accordance, for example, with characteristics of a particular user device/process stored intermittently or more persistently in storage 404. Linking engine 405 can also provide for restoring a Downloadable, such as a compressed, encrypted or otherwise encoded file that has been decompressed, decrypted or otherwise decoded via detection processing (e.g. see FIG. 6*b*).

It is discovered, for example, that the manner in which the Windows OS initiates a binary executable or an ActiveX control can be utilized to enable protected initiation of a detected-Downloadable. Linking engine 405 is, for example, configurable to form, for an ordinary single-executable Downloadable (e.g. an application program, applet, etc.) a sandboxed package 340 as a concatenation of ordered elements including an MPC 341, applicable policies 342 and the Downloadable or "XEQ" 343 (e.g. see FIG. 4).

Linking engine 405 is also configurable to form, for a Downloadable received by a server as a compressed single or multiple-executable Downloadable such as a zipped or meta file, a protecting package 340 including one or more MPCs, applicable policies and the one or more included executables of the Downloadable. For example, a sandboxed package can be formed in which a single MPC and policies precede and thus will affect all such executables as a result of inflating and installation. An MPC and applicable policies can also, for example, precede each executable, such that each executable will be separately sandboxed in the same or a different manner according to MPC/policy configuration (see above) upon inflation and installation. (See also FIGS. 5 and 6)

Linking engine is also configurable to form an initial MPC, MPC-policy or sandboxed package (e.g. prior to upon receipt of a downloadable) or an additional MPC, MPC-policy or sandboxed package (e.g. upon or following receipt of a downloadable), such that suitable MPCs/policies can be provided to a Downloadable-destination or other destination in a more distributed manner. In this way, requisite bandwidth or destination resources can be minimized (via two or more smaller packages) in compromise with latency or other considerations raised by the additional required communication.

A configurable linking engine can also be utilized in accordance with other requirements of particular devices/processes, further or different elements or other permutations in accordance with the teachings herein. (It might, for example be desirable to modify the ordering of elements, to provide one or more elements separately, to provide additional information, such as a header, etc., or perform other processing in accordance with a particular device, protocol or other application considerations.)

Policy/authentication reader-analyzer 481 summarily depicts other protection mechanisms that might be utilized in conjunction with Downloadable detection, such as already discussed, and that can further be configurable to operate in accordance with policies or parameters (summarily depicted by security/authentication policies 482). Integration of such further protection in the depicted configuration, for example, enables a potential-Downloadable from a known unfriendly source, a source failing authentication or a provided-source that is confirmed to be fictitious to be summarily discarded, otherwise blocked, flagged, etc. (with or without further processing). Conversely, a potential-Downloadable from a known friendly source (or one confirmed as such) can be transferred with or without further processing in accordance with particular application considerations. (Other configurations including pre or post Downloadable detection mechanisms might also be utilized.)

Finally, transfer engine 406 provides for receiving and causing linking engine 405 (or other protection) results to be transferred to a destination user device/process. As depicted, transfer engine 406 is configured to receive and transfer a Downloadable, a determined non-executable or a sandboxed package. However, transfer engine 406 can also be provided in a more configurable manner, such as was already discussed for other system 400 elements. (Any one or more of system 400 elements might be configurably implemented in accordance with a particular application.) Transfer engine 406 can perform such transfer, for example, by adding the information to a server transfer queue (not shown) or utilizing another suitable method.

Turning to FIG. 5 with reference to FIG. 4, a code detector 421 example is illustrated in accordance with an embodiment of the invention. As shown, code detector 421 includes data fetcher 501, parser 502, file-type detector 503, inflator 504 and control 506; other depicted elements. While implementable and potentially useful in certain instances, are found to require substantial overhead, to be less accurate in certain instances (see above) and are not utilized in a present implementation; these will be discussed separately below. Code detector elements are further configurable in accordance with stored parameters retrievable by data fetcher 501. (A coupling between data fetcher 501 and control 506 has been removed for clarity sake.)

Data fetcher 501 provides for retrieving a potential-Downloadable or portions thereof stored in buffer 407 or parameters from storage 404, and communicates such information or parameters to parser 502. Parser 502 receives a

potential-Downloadable or portions thereof from data fetcher 501 and isolates potential-Downloadable elements, such as file headers, source, destination, certificates, etc. for use by further processing elements.

File type detector 502 receives and determines whether the potential-Downloadable (likely) is or includes an executable file type. File-reader 502 can, for example, be configured to analyze a received potential-Downloadable for a file header, which is typically included in accordance with conventional data transfer protocols, such as a portable executable or standard ".exe" file format for Windows OS application programs, a Java class header for Java applets, and so on for other applications, distributed components, etc. "Zipped", meta or other compressed files, which might include one or more executables, also typically provide standard single or multi-level headers that can be read and used to identify included executable code (or other included information types). File type detector 502 is also configurable for analyzing potential-Downloadables for all potential file type delimiters or a more limited subset of potential file type delimiters (e.g. ".exe" or ".com" in conjunction with a DOS or Microsoft Windows OS Downloadable-destination).

Known file type delimiters can, for example, be stored in a more temporary or more persistent storage (e.g. storage 404 of FIG. 4) which file type detector 502 can compare to a received potential-Downloadable. (Such delimiters can thus also be updated in storage 404 as a new file type delimiter is provided, or a more limited subset of delimiters can also be utilized in accordance with a particular Downloadable-destination or other considerations of a particular application.) File type detector 502 further transfers to controller 506 a detected file type indicator indicating that the potential-Downloadable includes or does not include (i.e. or likely include) an executable file type.

In this example, the aforementioned detection processor is also included as predetection processor or, more particularly, a configurable file inflator 504. File inflator 504 provides for opening or "inflating" compressed files in accordance with a compressed file type received from file type detector 503 and corresponding file opening parameters received from data fetcher 501. Where a compressed file (e.g. a meta file) includes nested file type information not otherwise reliably provided in an overall file header or other information, inflator 504 returns such information to parser 502. File inflator 504 also provides any now-accessible included executables to control 506 where one or more included files are to be separately packaged with an MPC or policies.

Control 506, in this example, operates in accordance with stored parameters and provides for routing detected non-Downloadables or Downloadables and control information, and for conducting the aforementioned distributed downloading of packages to Downloadable-destinations. In the case of a non-Downloadable, for example, control 506 sends the non-Downloadable to transfer engine 406 (FIG. 4) along with any indicators that might apply. For an ordinary single-executable Downloadable, control 506 sends control information to agent generator 431 and the Downloadable to linking engine 405 along with any other applicable indicators (see 641 of FIG. 6b). Control 506 similarly handles a compressed single-executable Downloadable or a multiple downloadable to be protected using a single sandboxed package. For a multiple-executable Downloadable, control 506 sends control information for each corresponding executable to agent generator agent generator 431, and sends the executable to linking engine 405 along with controls and any applicable indicators, as in 643b of FIG. 6b. (The above assumes, however, that distributed downloading is not utilized; when used—according to applicable parameters—control 506 also operates in accordance with the following.)

Control 506 conducts distributed protection (e.g. distributed packaging) by providing control signals to agent generator 431, linking engine 405 and transfer engine 406. In the present example, control 506 initially sends controls to agent generator 431 and linking engine 405 (FIG. 4) causing agent generator to generate an initial MPC and initial policies, and sends control and a detected-Downloadable to linking engine 405. Linking engine 405 forms an initial sandboxed package, which transfer engine causes (in conjunction with further controls) to be downloaded to the Downloadable destination (643a of FIG. 6b). An initial MPC within the sandboxed package includes an installer and a communicator and performs installation as indicated below. The initial MPC also communicates via the communicator controls to control 506 (FIG. 5) in response to which control 506 similarly causes generation of MPC-M and policy-M modules 643c, which linking engine 405 links and transfer engine 406 causes to be sent to the Downloadable destination, and so on for any further such modules.

(It will be appreciated, however, that an initial package might be otherwise configured or sent prior to receipt of a Downloadable in accordance with configuration parameters or user interaction. Information can also be sent to other user devices, such as that of an administrator. Further MPCs/policies might also be coordinated by control 506 or other elements, or other suitable mechanisms might be utilized in accordance with the teachings herein.)

Regarding the remaining detection engine elements illustrated in FIG. 5, where content analysis is utilized, parser 502 can also provide a Downloadable or portions thereof to content detector 505. Content detector 505 can then provide one or more content analyses. Binary detector 551, for example, performs detection of binary information; pattern detector 552 further analyzes the Downloadable for patterns indicating executable code, or other detectors can also be utilized. Analysis results therefrom can be used in an absolute manner, where a first testing result indicating executable code confirms Downloadable detection, which result is then sent to control 506. Alternatively, however, composite results from such analyses can also be sent to control 506 for evaluation. Control 506 can further conduct such evaluation in a summary manner (determining whether a Downloadable is detected according to a majority or minimum number of indicators), or based on a weighting of different analysis results. Operation then continues as indicated above. (Such analysis can also be conducted in accordance with aspects of a destination user device or other parameters.)

FIG. 6a illustrates more specific examples of indicators/parameters and known (or "knowledge base") elements that can be utilized to facilitate the above-discussed system 400 configurability and detection. For clarity sake, indicators, parameters and knowledge base elements are combined as indicated "parameters." It will be appreciated, however, that the particular parameters utilized can differ in accordance with a particular application, and indicators, parameters or known elements, where utilized, can vary and need not correspond exactly with one another. Any suitable explicit or referencing list, database or other storage structure(s) or storage structure configuration(s) can also be utilized to implement a suitable user/device based protection scheme, such as in the above examples, or other desired protection schema.

Executable parameters 601 comprise, in accordance with the above examples, executable file type parameters 611,

executable code parameters 612 and code pattern parameters 613 (including known executable file type indicators, header/code indicators and patterns respectively, where code patterns are utilized). Use parameters 602 further comprise user parameters 621, system parameters 622 and general parameters 623 corresponding to one or more users, user classifications, user-system correspondences or destination system, device or processes, etc. (e.g. for generating corresponding MPCs/policies, providing other protection, etc.). The remaining parameters include interface parameters 631 for providing MPC/policy (or further) configurability in accordance with a particular device or for enabling communication with a device user (see below), and other parameters 632.

FIG. 6*b* illustrates a linking engine 405 according to an embodiment of the invention. As already discussed, linking engine 405 includes a linker for combining MPCs, policies or agents via concatination or other suitable processing in accordance with an OS, JVM or other host executor or other applicable factors that might apply. Linking engine 405 also includes the aforementioned post-detection processor which, in this example, comprises a compressor 508. As noted, compressor 508 receives linked elements from linker 507 and, where a potential-Downloadable corresponds to a compressed file that was inflated during detection, re-forms the compressed file. (Known file information can be provided via configuration parameters, substantially reversal of inflating or another suitable method.) Encryption or other post-detection processing can also be conducted by linking engine 508.

FIGS. 7*a*, 7*b* and 8 illustrate a "sandbox protection" system, as operable within a receiving destination-device, according to an embodiment of the invention.

Beginning with FIG. 7*a*, a client 146 receiving sandbox package 340 will "recognize" sandbox package 340 as a (mobile) executable and cause a mobile code installer 711 (e.g. an OS loader, JVM, etc.) to be initiated. Mobile code installer 711 will also recognize sandbox package 340 as an executable and will attempt to initiate sandbox package 340 at its "beginning." Protection engine 400 processing corresponding to destination 700 use of a such a loader, however, will have resulted in the "beginning" of sandbox package 340 as corresponding to the beginning of MPC 341, as noted with regard to the above FIG. 4 example.

Such protection engine processing will therefore cause a mobile code installer (e.g. OS loader 711, for clarity sake) to initiate MPC 341. In other cases, other processing might also be utilized for causing such initiation or further protection system operation. Protection engine processing also enables MPC 341 to effectively form a protection "sandbox" around Downloadable (e.g. detected-Downloadable or "XEQ") 343, to monitor Downloadable 343, intercept determinable Downloadable 343 operation (such as attempted accesses of Downloadable 343 to destination resources) and, if "malicious", to cause one or more other operations to occur (e.g. providing an alert, offloading the Downloadable, offloading the MPC, providing only limited resource access, possibly in a particular address space or with regard to a particularly "safe" resource or resource operation, etc.).

MPC 341, in the present OS example, executes MPC element installation and installs any policies, causing MPC 341 and protection policies 342 to be loaded into a first memory space, P1. MPC 341 then initiates loading of Downloadable 343. Such Downloadable initiation causes OS loader 711 to load Downloadable 343 into a further working memory space-P2 703 along with an API import table ("IAT") 731 for providing Downloadable 631 with destination resource access capabilities. It is discovered, however that the IAT can be modified so that any call to an API can be redirected to a function within the MPC. The technique for modifying the IAT is documented within the MSDN (Microsoft Developers Network) Library CD in several articles. The technique is also different for each operating system (e.g. between Windows 9x and Windows NT), which can be accommodated by agent generator configurability, such as that given above. MPC 341 therefore has at least initial access to API IAT 731 of Downloadable 632, and provides for diverting, evaluating and responding to attempts by Downloadable 632 to utilize system APIs 731, or further in accordance with protection policies 342. In addition to API diverting, MPC 341 can also install filter drivers, which can be used for controlling access to resources such as a Downloadable-destination file system or registry. Filter driver installation can be conducted as documented in the MSDN or using other suitable methods.

Turning to FIG. 8 with reference to FIG. 7*b*, an MPC 341 according to an embodiment of the invention includes a package extractor 801, executable installer 802, sandbox engine installer 803, resource access diverter 804, resource access (attempt) analyzer 805, policy enforcer 806 and MPC de-installer 807. Package extractor 801 is initiated upon initiation of MPC 341, and extracts MPC 341 elements and protection policies 342. Executable installer 802 further initiates installation of a Downloadable by extracting the downloadable from the protected package, and loading the process into memory in suspended mode (so it only loads into memory, but does not start to run). Such installation further causes the operating system to initialize the Downloadable's IAT 731 in the memory space of the downloadable process, P2, as already noted.

Sandbox engine installer 803 (running in process space P1) then installs the sandbox engine (803–805) and policies 342 into the downloadable process space P2. This is done in different way in each operating system (e.g. see above). Resource access diverter 804 further modifies those Downloadable-API IAT entries that correspond with protection policies 342, thereby causing corresponding Downloadable accesses via Downloadable-API IAT 731 to be diverted resource access analyzer 805.

During Downloadable operation, resource access analyzer or "RAA" 805 receives and determines a response to diverted Downloadable (i.e. "malicious") operations in accordance with corresponding protection policies of policies 342. (RAA 805 or further elements, which are not shown, can further similarly provide for other security mechanisms that might also be implemented.) Malicious operations can for example include, in a Windows environment: file operations (e.g. reading, writing, deleting or renaming a file), network operations (e.g. listen on or connect to a socket, send/receive data or view intranet), OS registry or similar operations (read/write a registry item), OS operations (exit OS/client, kill or change the priority of a process/thread, dynamically load a class library), resource usage thresholds (e.g. memory, CPU, graphics), etc.

Policy enforcer 806 receives RAA 805 results and causes a corresponding response to be implemented, again according to the corresponding policies. Policy enforcer 806 can, for example, interact with a user (e.g. provide an alert, receive instructions, etc.), create a log file, respond, cause a response to be transferred to the Downloadable using "dummy" or limited data, communicate with a server or other networked device (e.g. corresponding to a local or remote administrator), respond more specifically with a better known Downloadable, verify accessibility or user/

system information (e.g. via local or remote information), even enable the attempted Downloadable access, among a wide variety of responses that will become apparent in view of the teachings herein.

The FIG. **9** flowchart illustrates a protection method according to an embodiment of the invention. In step **901**, a protection engine monitors the receipt, by a server or other re-communicator of information, and receives such information intended for a protected information-destination (i.e. a potential-Downloadable) in step **903**. Steps **905–911** depict an adjunct trustworthiness protection that can also be provided, wherein the protection engine determines whether the source of the received information is known to be "unfriendly" and, if so, prevents current (at least unaltered) delivery of the potential-Downloadable and provides any suitable alerts. (The protection engine might also continue to perform Downloadable detection and nevertheless enable delivery or protected delivery of a non-Downloadable, or avoid detection if the source is found to be "trusted", among other alternatives enabled by the teachings herein.)

If, in step **913**, the potential-Downloadable source is found to be of an unknown or otherwise suitably authenticated/certified source, then the protection engine determines whether the potential-Downloadable includes executable code in step **915**. If the potential-Downloadable does not include executable code, then the protection engine causes the potential-Downloadable to be delivered to the information-destination in its original form in step **917**, and the method ends. If instead the potential-Downloadable is found to include executable code in step **915** (and is thus a "detected-Downloadable"), then the protection engine forms a sandboxed package in step **919** and causes the protection agent to be delivered to the information-Destination in step **921**, and the method ends. As was discussed earlier, a suitable protection agent can include mobile protection code, policies and the detected-Downloadable (or information corresponding thereto).

The FIG. **10***a* flowchart illustrates a method for analyzing a potential-Downloadable, according to an embodiment of the invention. As shown, one or more aspects can provide useful indicators of the inclusion of executable code within the potential-Downloadable. In step **1001**, the protection engine determines whether the potential-Downloadable indicates an executable file type, for example, by comparing one or more included file headers for file type indicators (e.g. extensions or other descriptors). The indicators can be compared against all known file types executable by all protected Downloadable destinations, a subset, in accordance with file types executable or desirably executable by the Downloadable-destination, in conjunction with a particular user, in conjunction with available information or operability at the destination, various combinations, etc.

Where content analysis is conducted, in step **1003** of FIG. **10***a*, the protection engine analyzes the potential-Downloadable and determines in accordance therewith whether the potential-Downloadable does or is likely to include binary information, which typically indicates executable code. The protection engine further analyzes the potential-Downloadable for patterns indicative of included executable code in step **1003**. Finally, in step **1005**, the protection engine determines whether the results of steps **1001** and **1003** indicate that the potential-Downloadable more likely includes executable code (e.g. via weighted comparison of the results with a suitable level indicating the inclusion or exclusion of executable code). The protection engine, given a suitably high confidence indicator of the inclusion of executable code, treats the potential-Downloadable as a detected-Downloadable.

The FIG. **10***b* flowchart illustrates a method for forming a sandboxed package according to an embodiment of the invention. As shown, in step **1011**, a protection engine retrieves protection parameters and forms mobile protection code according to the parameters. The protection engine further, in step **1013**, retrieves protection parameters and forms protection policies according to the parameters. Finally, in step **1015**, the protection engine couples the mobile protection code, protection policies and received-information to form a sandboxed package. For example, where a Downloadable-destination utilizes a standard windows executable, coupling can further be accomplished via concatenating the MPC for delivery of MPC first, policies second, and received information third. (The protection parameters can, for example, include parameters relating to one or more of the Downloadable destination device/process, user, supervisory constraints or other parameters.)

The FIG. **11** flowchart illustrates how a protection method performed by mobile protection code ("MPC") according to an embodiment of the invention includes the MPC installing MPC elements and policies within a destination device in step **1101**. In step **1102**, the MPC loads the Downloadable without actually initiating it (i.e. for executables, it will start a process in suspended mode). The MPC further forms an access monitor or "interceptor" for monitoring or "intercepting" downloadable destination device access attempts within the destination device (according to the protection policies in step **1103**, and initiates a corresponding Downloadable within the destination device in step **1105**.

If, in step **1107**, the MPC determines, from monitored/intercepted information, that the Downloadable is attempting or has attempted a destination device access considered undesirable or otherwise malicious, then the MPC performs steps **1109** and **1111**; otherwise the MPC returns to step **1107**. In step **1109**, the MPC determines protection policies in accordance with the access attempt by the Downloadable, and in step **1111**, the MPC executes the protection policies. (Protection policies can, for example, be retrieved from a temporary, e.g. memory/cache, or more persistent storage.)

As shown in the FIG. **12***a* example, the MPC can provide for intercepting Downloadable access attempts by a Downloadable by installing the Downloadable (but not executing it) in step **1201**. Such installation will cause a Downloadable executor, such as a the Windows operating system, to provide all required interfaces and parameters (such as the IAT, process ID, etc.) for use by the Downloadable to access device resources of the host device. The MPC can thus cause Downloadable access attempts to be diverted to the MPC by modifying the Downloadable IAT, replacing device resource location indicators with those of the MPC (step **1203**).

The FIG. **12***b* example further illustrates an example of how the MPC can apply suitable policies in accordance with an access attempt by a Downloadable. As shown, the MPC receives the Downloadable access request via the modified IAT in step **1211**. The MPC further queries stored policies to determine a policy corresponding to the Downloadable access request in step **1213**.

The foregoing description of preferred embodiments of the invention is provided by way of example to enable a person skilled in the art to make and use the invention, and in the context of particular applications and requirements thereof. Various modifications to the embodiments will be readily apparent to those skilled in the art, and the generic principles defined herein may be applied to other embodi-

ments and applications without departing from the spirit and scope of the invention. Thus, the present invention is not intended to be limited to the embodiments shown, but is to be accorded the widest scope consistent with the principles, features and teachings disclosed herein. The embodiments described herein are not intended to be exhaustive or limiting. The present invention is limited only by the following claims.

What is claimed is:

1. A processor-based method, comprising:

receiving downloadable-information;

determining whether the downloadable-information includes executable code; and

causing mobile protection code to be communicated to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code,

wherein the determining comprises performing one or more analyses of the downloadable-information, the analyses producing detection-indicators indicating whether a correspondence is detected between a downloadable-information characteristic and at least one respective executable code characteristic, and evaluating the detection-indicators to determine whether the downloadable-information includes executable code.

2. The method of claim 1, wherein at least one of the detection-indicators indicates a level of downloadable-information characteristic and executable code characteristic correspondence.

3. The method of claim 1, wherein the evaluating includes assigning a weighted level of importance to at least one of the indicators.

4. A processor-based method, comprising:

receiving downloadable-information;

determining whether the downloadable-information includes executable code; and

causing mobile protection code to be communicated to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code,

wherein the causing mobile protection code to be communicated comprises forming a sandboxed package including the mobile protection code and the downloadable-information, and causing the sandboxed package to be communicated to the at least one information-destination.

5. The method of claim 4, wherein the sandboxed package is formed such that the mobile protection code will be executed by the information-destination before the downloadable-information.

6. The method of claim 5, wherein the sandboxed package further includes protection policies according to which the mobile protection code is operable.

7. The method of claim 6, wherein the sandboxed package is formed for receipt by the information-destination such that the mobile protection code is received before the downloadable-information, and the downloadable information before the protection policies.

8. The method of claim 6, wherein the protection policies correspond with at least one of the information-destination and a user of the information destination.

9. A processor-based system, comprising:

an information monitor for receiving downloadable-information;

a content inspection engine communicatively coupled to the information monitor for determining whether the downloadable-information includes executable code; and

a packaging engine communicatively coupled to the content inspection engine for causing mobile protection code ("MPC") to be communicated to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code,

wherein the content inspection engine comprises one or more downloadable-information analyzers for analyzing the downloadable-information, each analyzer producing therefrom a detection indicator indicating whether a downloadable-information characteristic corresponds with an executable code characteristic, and an inspection controller communicatively coupled to the analyzers for determining whether the indicators indicate that the downloadable-information includes executable code.

10. The system of claim 9, wherein at least one of the detection-indicators indicates a level of downloadable-information characteristic and executable code characteristic correspondence.

11. The system of claim 9, wherein the evaluating includes assigning a weighted level of importance to at least one of the detection-indicators.

12. A processor-based system, comprising:

an information monitor for receiving downloadable-information;

a content inspection engine communicatively coupled to the information monitor for determining whether the downloadable-information includes executable code; and

a packaging engine communicatively coupled to the content inspection engine for causing mobile protection code ("MPC") to be communicated to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code,

wherein the packaging engine comprises an MPC generator for providing the MPC, a linking engine coupled to the MPC generator for forming a sandbox package including the MPC and the downloadable-information, and a transfer engine for causing the sandbox package to be communicated to the at least one information-destination.

13. The system of claim 12, wherein the packaging engine further comprises a policy generator communicatively coupled to the linking engine for providing protection policies according to which the MPC is operable.

14. The system of claim 13, wherein the sandboxed package is formed for receipt by the information-destination such that the mobile protection code is executed before the downloadable-information.

15. The system of claim 14, wherein the protection policies correspond with policies of at least one of the information-destination and a user of the information destination.

16. A processor-based method, comprising:

receiving, at an information re-communicator, downloadable-information, including executable code; and

causing mobile protection code to be executed by a mobile code executor at a downloadable-information destination such that one or more operations of the executable code at the destination, if attempted, will be processed by the mobile protection code,

wherein the causing is accomplished by forming a sandboxed package including the mobile protection code and the downloadable-information, and causing the sandboxed package to be delivered to the downloadable-information destination.

17. The method of claim 16, wherein the sandboxed package further includes protection policies according to which the processing by the mobile protection code is conducted.

18. A sandboxed package formed according to the method of claim 17.

19. The method of claim 17, wherein the forming comprises generating the mobile protection code, generating the sandboxed package, and linking the mobile protection code, protection policies and downloadable-information.

20. The method of claim 19, wherein the generating of at least one of the mobile protection code and the protection policies is conducted in accordance with one or more destination-characteristics of the destination.

21. The method of claim 20, wherein the destination-characteristics include characteristics corresponding to at least one of a destination user, a destination device and a destination process.

22. A sandboxed package formed according to the method of claim 16.

23. The method of claim 16, wherein the causing the sandboxed package to be executed includes communicating the sandboxed package to a communication buffer of the information re-communicator.

24. The method of claim 16, wherein the re-communicator is at least one of a firewall and a network server.

25. The method of claim 16, wherein the sandboxed package has a same file type as the downloadable-information, thereby causing the mobile code executor to be unaware that the protected package is not a normal downloadable.

26. The method of claim 25, wherein the sandboxed package is formed using concatenation of a mobile protection code, a policy, and a downloadable.

27. The method of claim 16, wherein executing the mobile protection code at the destination causes downloadable interfaces to resources at the destination to be modified such that at least one attempted operation of the executable code is diverted to the mobile protection code.

28. A processor-based system, comprising:

receiving means for receiving, at an information re-communicator, downloadable-information, including executable code; and

mobile code means communicatively coupled to the receiving means for causing mobile protection code to be executed by a mobile code executor at a downloadable-information destination such that one or more operations of the executable code at the destination, if attempted, will be processed by the mobile protection code,

wherein the causing is accomplished by forming a sandboxed package including the mobile protection code and the downloadable-information, and causing the sandboxed package to be delivered to the downloadable-information destination.

29. The system of claim 28, wherein the sandboxed package further includes protection policies according to which the processing by the mobile protection code is conducted.

30. The system of claim 29, wherein the forming comprises generating the mobile protection code, generating the protection policies, and linking the mobile protection code, protection policies and downloadable-information.

31. The system of claim 30, wherein the generating of at least one of the mobile protection code and the protection policies is conducted in accordance with one or more destination-characteristics of the destination.

32. The system of claim 31, wherein the destination-characteristics include characteristics corresponding to at least one of a destination user, a destination device and a destination process.

33. The system of claim 28, wherein the causing the sandboxed package to be executed includes communicating the sandboxed package to a communication buffer of the information re-communicator.

34. The system of claim 33, wherein the re-communicator is at least one of a firewall and a network server.

35. The system of claim 34, wherein executing the mobile protection code at the destination causes downloadable interfaces a resource at the destination to be modified such that at least one attempted operation of the executable code is diverted to the mobile protection code.

* * * * *

US006357010B1

(12) **United States Patent**
Viets et al.

(10) **Patent No.: US 6,357,010 B1**
(45) **Date of Patent: Mar. 12, 2002**

(54) **SYSTEM AND METHOD FOR CONTROLLING ACCESS TO DOCUMENTS STORED ON AN INTERNAL NETWORK**

(75) Inventors: **Richard R. Viets, Naples; David G. Motes, Bonita Springs, both of FL (US); Paula Budig Greve, St. Anthony; Wayne W. Herberg, Rush City, both of MN (US)**

(73) Assignee: **Secure Computing Corporation, Roseville, MN (US)**

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/024,576**

(22) Filed: **Feb. 17, 1998**

(51) Int. Cl.[7] G06F 12/14; G06F 15/173; H04L 12/00; H04L 9/00

(52) U.S. Cl. 713/201; 709/225; 713/200

(58) Field of Search 713/201, 200; 709/225

(56) **References Cited**

U.S. PATENT DOCUMENTS

3,956,615	A	5/1976	Anderson et al.	235/61.7 B
4,177,510	A	12/1979	Appell et al.	364/200
4,584,639	A	4/1986	Hardy	364/200
4,621,321	A	11/1986	Boebert et al.	364/200
4,701,840	A	10/1987	Boebert et al.	364/200
4,713,753	A	12/1987	Boebert et al.	364/200
4,914,568	A	4/1990	Kodosky et al.	364/200
5,124,984	A	6/1992	Engle	370/94.1
5,179,658	A *	1/1993	Izawa	345/508
5,204,812	A *	4/1993	Kaniaj et al.	707/9
5,272,754	A	12/1993	Boebert	380/25
5,276,735	A	1/1994	Boebert et al.	380/21
5,311,593	A	5/1994	Carmi	380/23
5,329,623	A	7/1994	Smith et al.	395/275
5,455,953	A *	10/1995	Russell	710/266
5,544,321	A *	8/1996	Theimer et al.	714/9
5,566,170	A	10/1996	Bakke et al.	370/60
5,586,260	A	12/1996	Hu	395/200.2
5,606,668	A	2/1997	Shwed	395/200.11
5,619,648	A	4/1997	Canale et al.	395/200.01

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

EP	0697662 A1	2/1996	G06F/12/14
EP	0 743 777 A2	11/1996	H04L/29/06
EP	0811939 A2	12/1997	G06F/17/30
WO	97/13340	4/1997	H04L/9/00
WO	97/16911	5/1997	H04L/29/06
WO	97/26731	7/1997	H04L/9/00

OTHER PUBLICATIONS

Yialelis et al. "Role–Based Security for Distributed Object Systems", IEEE Proceeding, 1996, pp. 80–85.*
Sandhu et al. "Role–Based Access Control Models", IEEE Computer, Feb. 1996, pp. 38–47.*
Tari et al. "Role–Based Access Control For Intranet Security", IEEE Internet Computing, 1997, pp. 24–34.*

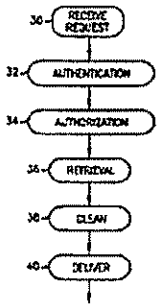
(List continued on next page.)

Primary Examiner—Thomas Lee
Assistant Examiner—Tanh Q. Nguyen
(74) *Attorney, Agent, or Firm*—Schwegman, Lundberg, Woessner & Kluth, P.A.

(57) **ABSTRACT**

A system and method of limiting access from an external network to documents stored on an internal network. A client list is built in which each client is assigned to one or more roles. Each role has access to one or more documents as defined on a document list. A request from an external network is reviewed and, if possible, the request is associated with a client on the client list. The requested document is then compared to the document list associated with the client's role and, if the requested document is in the list of documents available to a client in the client's role, the requested document is fetched, cleaned and sent to the client.

37 Claims, 6 Drawing Sheets



U.S. PATENT DOCUMENTS

5,623,601	A		4/1997	Vu 395/187.01
5,636,371	A		6/1997	Yu 395/500
5,673,322	A		9/1997	Pope et al. 380/49
5,684,951	A		11/1997	Goldman et al. 395/188.01
5,689,566	A		11/1997	Nguyen 380/25
5,701,137	A	*	12/1997	Kiernan et al. 345/340
5,708,780	A		1/1998	Levergood et al. 395/200.12
5,784,566	A	*	7/1998	Viswanathan et al. 709/229
5,802,299	A	*	9/1998	Logan et al. 709/218
5,819,271	A	*	10/1998	Mahoney et al. 707/9
5,826,029	A	*	10/1998	Gore, Jr. et al. 709/227
5,864,683	A	*	1/1999	Boebert et al. 709/249
5,864,871	A	*	1/1999	Kitain et al. 707/104
5,870,544	A	*	2/1999	Curtis 713/201
5,884,033	A	*	3/1999	Duvall et al. 709/206
5,884,312	A	*	3/1999	Dustan et al. 707/10
5,892,905	A	*	4/1999	Brandt et al. 713/201
5,903,732	A	*	5/1999	Reed et al. 709/229
5,911,143	A	*	6/1999	Deinhart et al. 707/103
5,913,024	A	*	6/1999	Green et al. 713/200
5,915,087	A	*	6/1999	Hammond et al. 713/201
5,918,013	A	*	6/1999	Mighdoll et al. 709/217
5,933,600	A	*	8/1999	Shieh et al. 709/219
5,950,195	A	*	9/1999	Stockwell et al. 707/4
5,961,601	A	*	10/1999	Iyengar 709/229
5,987,611	A	*	11/1999	Freund 713/201
6,023,765	A	*	2/2000	Kuhn 713/200
6,055,637	A	*	4/2000	Hudson et al. 713/201
6,088,679	A	*	7/2000	Barkley 705/8

OTHER PUBLICATIONS

International Search Report , PCT Application No. PCT/US 95/12681, 8 p. (mailed Apr. 9, 1996).

Ancilotti, P., et al., "Language Features for Access Control", *IEEE Transactions on Software Engineering, SE–9*, 16–25 (Jan. 1983).

Atkinson, R., "IP Authentication Header", Network Working Group, Request For Comment No. 1826, http//ds.internic.net/rfc/rfc1826.txt, 9 p. (Aug. 1995).

Atkinson, R., "IP Encapsulating Security Payload (ESP)", Network Working Group, Request For Comment No. 1827, http//ds.internic.net/rfc/rfc1827.txt, 12 p. (Aug. 1995).

Atkinson, R., "Security Architecture for the Internet Protocol", Network Working Group, Request for Comment No. 1825, http//ds.internic.net/rfc/rfc1825.txt, 21 p. (Aug. 1995).

Baclace, P.E., "Competitive Agents for Information Filtering", *Communications of the ACM*, 35, 50 (Dec. 1992).

Badger, L., et al., "Practical Domain and Type Enforcement for UNIX", *Proceedings of the 1995 IEEE Symposium on Security and Privacy*, p. 66–77 (May 1995).

Belkin, N.J., et al., "Information Filtering and Information Retrieval: Two Sides of the Same Coin?", *Communications of the ACM*, 35, 29–38 (Dec. 1992).

Bellovin, S.M., et al., "Network Firewalls", *IEEE Communications Magazine*, 32, 50–57 (Sep. 1994).

Bevier, W.R., et al., "Connection Policies and Controlled Interference", *Proceedings of the Eighth IEEE Computer Security Foundations Workshop*, Kenmare, Ireland, p. 167–176 (Jun. 13–15, 1995).

Bowen, T.F., et al., "The Datacycle Architecture", *Communications of the ACM*, 35, 71–81 (Dec. 1992).

Bryan, J., "Firewalls For Sale", *BYTE*, 99–100, 102, 104–105 (Apr. 1995).

Cobb, S., "Establishing Firewall Policy", *IEEE*, 198–205 (1996).

Foltz, P.W., et al., "Personalized Information Delivery: An Analysis of Information Filtering Methods", *Communications of the ACM*, 35, 51–60 (Dec. 1992).

Gassman, B., "Internet Security, and Firewalls Protection on the Internet", *IEEE*, 93–107 (1996).

Goldberg, D., et al., "Using Collaborative Filtering to Weave an Information Tapestry", *Communications of the ACM*, 35, 61–70 (DEC. 1992).

Grampp, F.T. "UNIX Operating System Security", *AT& T Bell Laboratories Technical Journal*, 63, 1649–1672 (Oct. 1984).

Greenwald, M., et al., "Designing an Academic Firewall: Policy, Practice, and Experience with SURF", *IEEE*, 79–92 (1996).

Haigh, J.T., et al., "Extending the Noninterference Version of MLS for SAT", Proceedings of the 1986 IEEE Symposium on Security and Privacy, Oakland, CA, p. 232–239 (Apr. 7–9, 1986).

Kam, P., et al., "The ESP DES–CBC Transform", Network Working Group, Request for Comment No. 1829, http// ds.internic.net/rfc/rfc1829.txt, 9 p. (Aug. 1995).

Kent, S.T., "Internet Privacy Enhanced Mail", *Communications of the ACM*, 36, 48–60 (Aug. 1993).

Lampson, B.W., et al., "Dynamic Protection Structures", *AFIPS Conference Proceedings*, 35, 1969 Fall Joint Computer Conference, Las Vegas, NV, 27–38 (Nov. 18–20, 1969).

Lee, K.C., et al., "A Framework for Controlling Cooperative Agents", *Computer*, 8–16 (Jul. 1993).

Lodin, S.W., et al., "Firewalls Fend Off Invasions from the Net", *IEEE Spectrum*, 26–34 (Feb. 1998).

Loeb, S., "Architecting Personalized Delivery of Multimedia Information", *Communications of the ACM*, 35, 39–48 (1992).

Loeb, S., et al., "Information Filtering", *Communications of the ACM*, 35, 26–28 (Dec. 1992).

Marenbloom, P., "Network 'Fire Walls' Safeguard LAN Data from Outside Intrusion", *Infoworld*, p. 69 & addnl page (Jul. 25, 1994).

Metzger, P., et al., "IP Authentication using Keyed MD5", Network Working Group, Request for Comments No. 1828, http//ds.internic.net/rfc/rfc1828.txt, 5 p. (Aug. 1995).

Obraczka, K., et al., "Internet Resource Discovery Services", *Computer*, 8–22 (Sep. 1993).

Peterson, L.L., et al., In: *Computer Networks*, Morgan Kaufmann Publishers, Inc., San Francisco, CA, p. 218–221, 284–286 (1996).

Press, L., "The Net: Progress and Opportunity", *Communications of the ACM*, 35, 21–25 (Dec. 1992).

Schroeder, M.D., et al., "A Hardware Architecture for Implementing Protection Rings", *Communications of the ACM*, 15, 157–170 (Mar. 1972).

Schwartz, M.F., "Internet Resource Discovery at the University of Colorado", *Computer*, 25–35 (Sep. 1993).

Smith, R.E., "Constructing a High Assurance Mail Guard", Secure Computing Corporation (Appeared in the Proceedings of the National Computer Security Conference), 7 p. (1994).

Smith, R.E., "Sidewinder: Defense in Depth Using Type Enforcement", *International Journal of Network Management*, p. 219–229 (Jul.–Aug. 1995).

Stadnyk, I., et al., "Modeling User's Interests in Information Filters", *Communications of the ACM*, 35, 49–50 (Dec. 1992).

Stempel, S., "IpAccess—An Internet Service Access System for Firewall Installations", *IEEE*, 31–41 (1995).
Stevens, C., "Automating the Creation of Information Filters", *Communications of the ACM*, 35, 48 (Dec. 1992).
Thomsen, D., "Type Enforcement: The New Security Model", *SPIE*, 2617, 143–150 (1995).
Warrier, U.S., et al., "A Platform for Heterogeneous Interconnection Network Management", *IEEE Journal on Selected Areas in Communications*, 8, 119–126 (Jan. 1990).
White, L.J., et al., "A Firewall Concept for Both Control-Flow and Data-Flow in Regression Integration Testing", *IEEE*, 262–271 (1992).
Wolfe, A., "Honeywell Builds Hardware for Computer Security", *Electronics*, 14–15 (Sep. 2, 1985).
Boebert, W.E., et al., "Secure Ada Target: Issues, System Design, and Verification", *Proceedings of the Symposium on Security and Privacy*, Oakland, California, Oakland, California, pp. 59–66, (1985).
Boebert, W.E., et al., "Secure Computing: The Secure Ada Target Approach", *Sci. Honeywellet*, 6(2), 17 pages, (1985).
Kahan, J., "A capability based authorization model for the world-Wide Web", *Computer Networks and ISDN Systems*, pp. 1055–1064, (1995).
Vinter, S.T., et al., "Extended Discretionary Access Controls", *IEEE*, pp. 39–49, (1988).

* cited by examiner

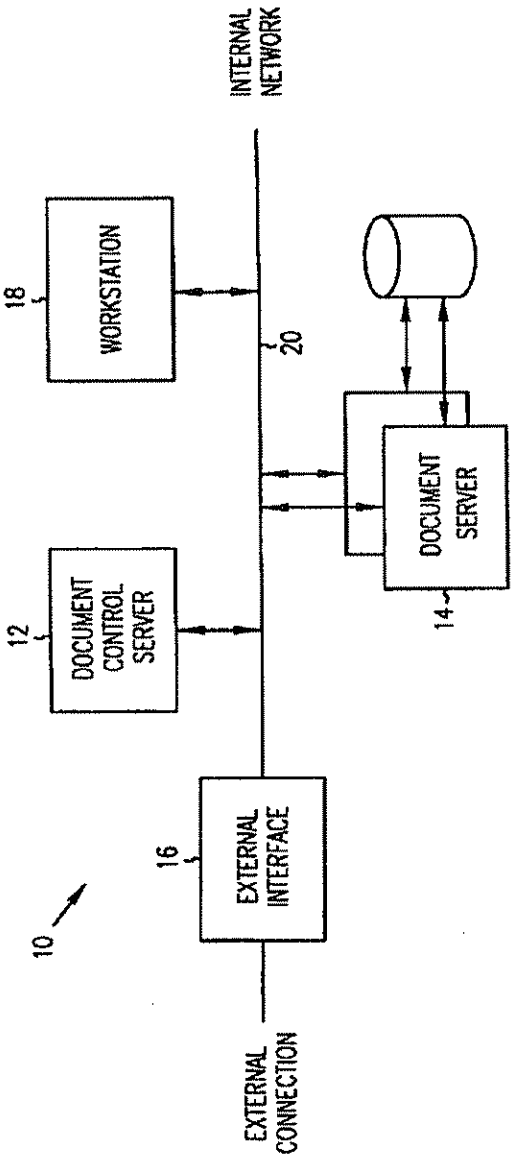

INTERNAL NETWORK
WORKSTATION
18
20
DOCUMENT SERVER
14
DOCUMENT CONTROL SERVER
12
EXTERNAL INTERFACE
16
10
EXTERNAL CONNECTION

FIG. 1

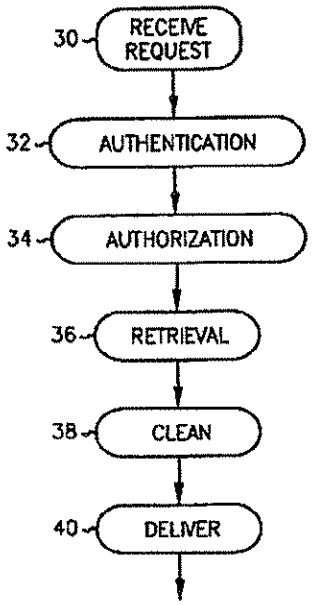

30
RECEIVE REQUEST
32
AUTHENTICATION
34
AUTHORIZATION
36
RETRIEVAL
38
CLEAN
40
DELIVER

FIG. 2

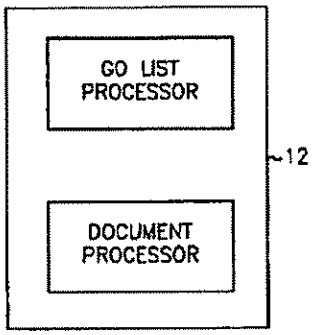

GO LIST PROCESSOR
DOCUMENT PROCESSOR
12

FIG. 3

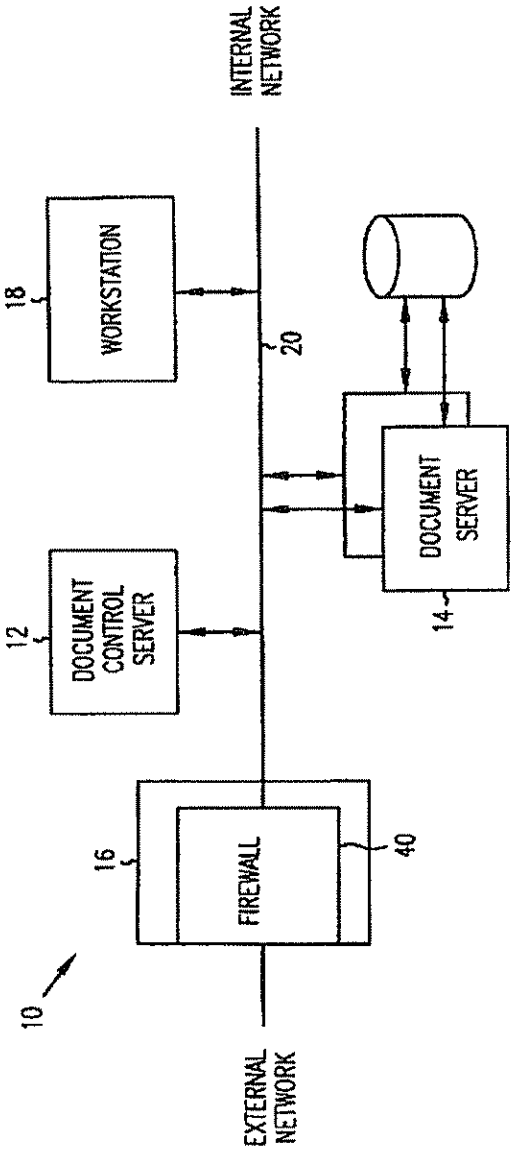

INTERNAL NETWORK
WORKSTATION
18
20
DOCUMENT SERVER
14
DOCUMENT CONTROL SERVER
12
FIREWALL
16
40
10
EXTERNAL NETWORK

FIG. 4

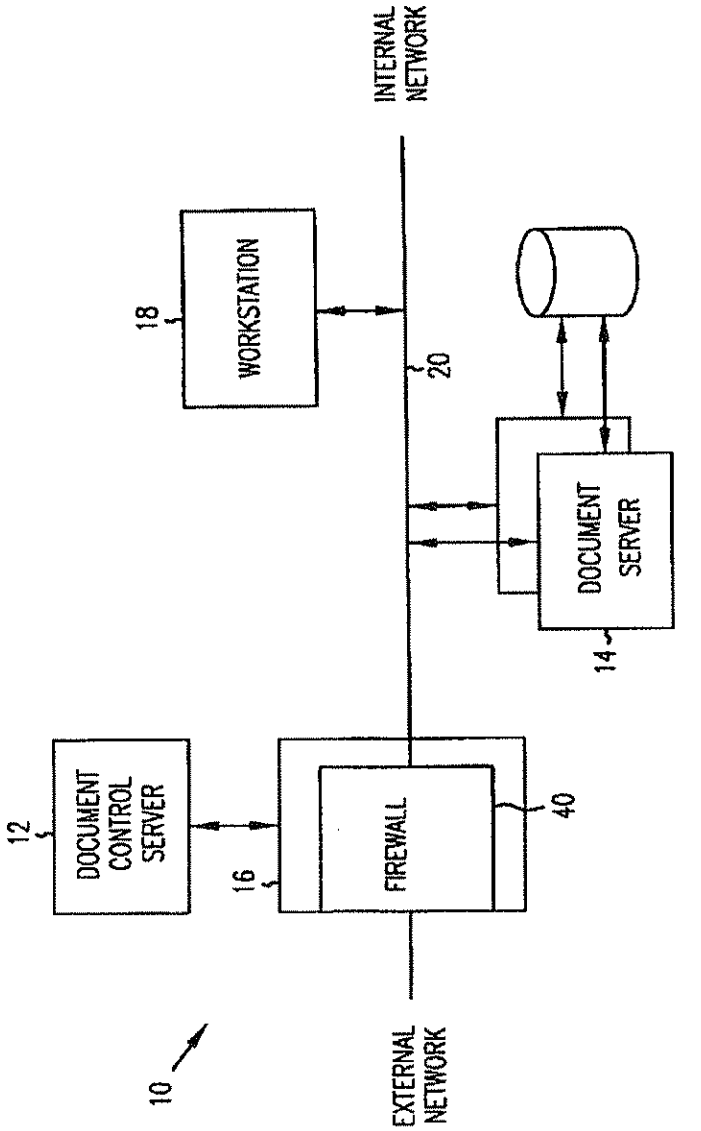

INTERNAL NETWORK
WORKSTATION
18
20
DOCUMENT SERVER
14
DOCUMENT CONTROL SERVER
12
16
FIREWALL
40
10
EXTERNAL NETWORK

FIG. 5

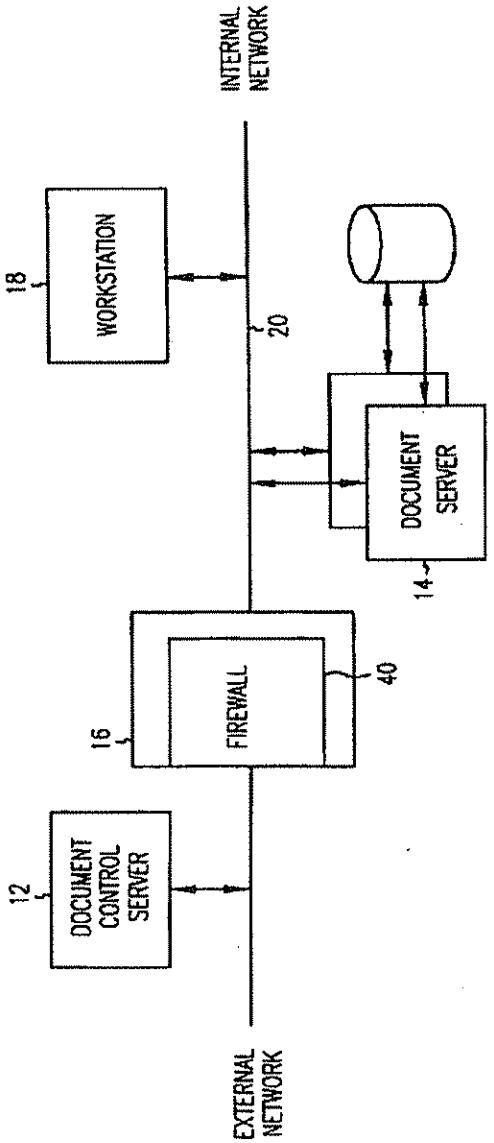

INTERNAL NETWORK
WORKSTATION
18
20
DOCUMENT SERVER
14
FIREWALL
16
40
DOCUMENT CONTROL SERVER
12
EXTERNAL NETWORK

FIG. 6

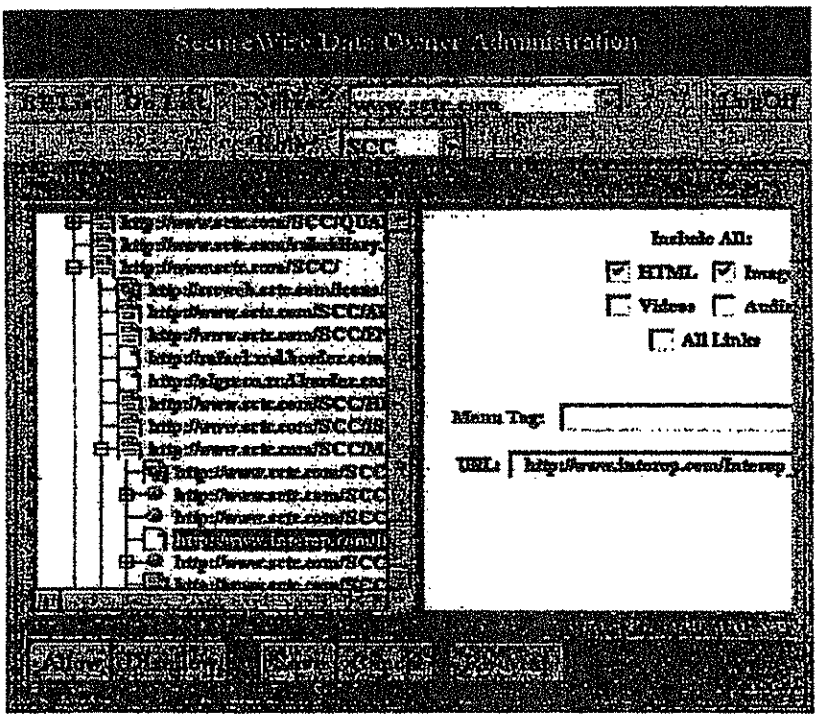


FIG. 7

SYSTEM AND METHOD FOR CONTROLLING ACCESS TO DOCUMENTS STORED ON AN INTERNAL NETWORK

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to systems and methods for controlling communication between networks, and in particular to a system and method for limiting access to documents stored on an internal network.

2. Background Information

Businesses today are acting cooperatively to achieve compatible business goals. For example, companies are using just-in-time manufacturing techniques to reduce overhead. To make this work, companies rely heavily on the ability of their suppliers to provide materials when needed.

At the same time, in this digital age business executives have become accustomed to receiving information from a number of sources both inside and outside the company almost instantaneously. They rely on such information to drive their day-to-day management decisions.

In order to provide outside organizations with relevant information in a timely manner, many companies have expanded their order-processing departments to handle increased call volumes. In this environment, outside partners call into the company's order-processing department to request specific information. This requires an employee to be available to answer calls, pull up information and verbally convey information to the partner. This option is very expensive, slow, and offers a poor level of service. What is needed is a system and method of streamlining the flow of information between partner companies while limiting access to company proprietary information.

The Internet provides one possible solution to this problem. The nature of the Internet makes it an ideal vehicle for organizations to communicate and share information. The Internet offers low cost universal access to information. Because of this, Internet transactions are expected to more than quadruple over the next two years, and partner communications via the Internet will almost double. Companies have began to look to the Internet as a medium allowing quick, easy and inexpensive to business partners. To date, however, their Internet options have been limited.

One solution is to give business partners access to the company internal network. Companies are hesitant to do this, however, since such access, if abused, can lead to the disclosure of company sensitive information.

Another solution is to replicate necessary information to a web server located outside the company'firewall. Such an approach does allow organizations direct access to the information while at the same time limiting their access to company sensitive information. For this environment to work, however, the MIS department must manually transfer information from the internal network to the external server. Therefore, while this option offers organizations direct access to necessary data, that information can be 24 to 48 hours old. When dealing with just-in-time inventory levels and large dollar amounts, 24 hours is too late. This option also creates a bottleneck in MIS, redundancy of data, and decreased data integrity.

What is needed is a system and method for giving controlled access to designated documents stored on the internal network while restricting access to company sensitive information.

SUMMARY OF THE INVENTION

The present invention is a system and method of limiting access from an external network to documents stored on an internal network. A client list is built in which each client is assigned to one or more roles. Each role has access to one or more documents as defined on a document list. A request from an external network is reviewed and, if possible, the request is associated with a client on the client list. The requested document is then compared to the document list associated with the client'role and, if the requested document is in the list of documents available to a client in the client's role, the requested document is fetched, cleaned and sent to the client.

According to another aspect of the present invention, a document control system is described. The document control system includes an internal network, an external interface, a document server connected to the internal network, and a document control server connected to the internal network and to the external interface. The document server controls access to a plurality of documents, including a first document. The document control server includes a go list processor for determining if the user has authorization to access said first document and a document processor for reading the first document from the document server, cleaning the first document and forwarding a clean version of said first document to the user. In operation, the document control server receives a document request from the external interface for the first document, determines a user associated with the document request, authenticates the user, determines if the user has authorization to access said first document and, if authorized, reads the first document from the document server, cleans the first document and forwards a clean version of said first document to the user.

BRIEF DESCRIPTION OF THE DRAWINGS

In the drawings, where like numerals refer to like components throughout the several views,

FIG. 1 shows a document access system;

FIG. 2 is a flow diagram illustrating operations performed by the document access system of FIG. 1;

FIG. 3 shows a document control server which can be used in the document access system shown in FIG. 1;

FIG. 4 is a document access system which includes a firewall;

FIG. 5 is a document access system in which the document control server is placed in a third network;

FIG. 6 is a document access system in which the document control server is placed on the external network; and

FIG. 7 is an example of a tree structure representation which could be used to aid the data owner in the selection of permitted URLs.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

In the following detailed description of the preferred embodiments, reference is made to the accompanying drawings which form a part hereof, and in which is shown by way of illustration specific embodiments in which the invention may be practiced. It is to be understood that other embodiments may be utilized and structural changes may be made without departing from the scope of the present invention.

As noted above, corporations today are required by customers to deliver information such as price changes, new product data, manufacturing data, and customer support electronically. Competition is driving firms to work with partners through tight connections to internal systems. Allowing access, however, to such data in an efficient, manageable, and secure manner presents challenges. Com-

panies go to great lengths to set up order processing departments and replicate large quantities of internal data to an external Internet server. These efforts are not only inefficient, but usually result in redundancy of data, decreased network integrity, and a bottleneck in the MIS department.

The present invention solves this problem by allowing specified external users controlled, customized, and secure access to the company's intranet without complex network infrastructure modifications. Further, the present invention permits one to control the parts of a Web server that are accessible to a Business Partner with only minimal intervention by IS personnel. (The term "Business Partner" is used in the following discussion to describe an external user who needs access to data such as Web pages which are not generally available to the public, but who also should not have unlimited to a company's intranet Web services.)

A document access system for giving controlled access to designated documents stored on the internal network while restricting access to company sensitive information is shown in FIG. 1. In FIG. 1 document access system 10 includes a document control server 12, a document server 14, an external interface 16 and one or more internal workstations 18. Document control server 12, document server 14, external interface 16 and internal workstations 18 are interconnected via internal network 20. Document server 14 reads and writes documents to storage 20. Requests for documents arrive at external interface 16 and are forwarded to document control server 12 for execution. In one embodiment external interface 16 includes a router used to form an Internet connection. In another embodiment, external interface 16 includes a direct connection interface such as formed by one or more modems used for direct dial-up by business partners wishing to access their data.

In one embodiment, as is illustrated in FIG. 2, at 30 document control server 12 receives a document request from the external interface for a first document. At 32, document control server 12 determines a user associated with the document request and authenticates the user. At 34, system 10 checks to see if the user is authorized to access the document requested. If so, at 36, system 10 retrieves the document from document server 14, cleans the document at 38 and, at 40, forwards the clean version of the document to the user. One embodiment of such a system and method is described next for a HyperText Transfer Protocol (HTTP) system.

When an HTTP or HTTPS connection request comes into document control server 12 there are three critical functions that must take place prior to returning the requested Web page: authentication, authorization, and internal connection. If either of the first two functions fail, the internal connection is not made. Then, once the internal connection has been made, document control server 12 must parse and "clean" the Web page prior to returning it to the requesting user.

Authentication

Authentication is fairly straight-forward and is of course visible to the end user. Following the HTTP protocol, when a user first enters a Uniform Resource Locator (URL) from their browser and the request is received at 30 (see FIG. 2) at server 12, a check is made at 32 for authentication. If basic authentication is being used, a check is made for authentication information in the HTTP header. If no username and password are found, the server returns a 401 error to the browser, telling the browser it needs to authenticate. The browser then pops up the box prompting the user to enter a username and password. When the HTTP request comes into the server, document control server 12 parses the username and password, comparing against its internal database of users and if it finds a match, lets control proceed onto the authorization step. If no match is found, document control server 12 returns an error code back to the Web server it is running under (e.g., Internet Information Server or Netscape Enterprise Server) and the server will once again send back a 401 requesting the username and password. This process can happen 3 times and then the server will deny access. In one embodiment this authentication check is checked against a data base of known users as opposed to letting the Web Server check against a user database it may have.

Authorization

Once document control server 12 has authenticated the request, it must, at 34, determine if the user is authorized to get to the URL they have requested. This authorization will fail if the URL the user requested is not in the list of "allowed" URLs associated with the user. In one embodiment, each user is assigned one or more roles. Each role has access to a set of allowed URLs associated with that role.

In one embodiment each user has one or more roles associated with their user ID. For instance, they could be in the Marketing role, as well as the Engineering role. In one such embodiment, each role is directly associated with an internal server; you can only define one role for each server. This means you could not have the Marketing role and the Engineering role going to the same physical internal server. Such an approach can simplify system design.

In another embodiment, more than one roles may be assigned to each internal server. For example, a manufacturer may have all his reseller information on one server. One role, however, contains international resellers and another role contains domestic resellers. In such an embodiment, it would be advantageous to be able to define different sets of URLs on a single document server 14 that would allow for the different roles.

To complete the authorization portion, document control server 12 scans the list of allowed URLs for each role the user is in until it finds a match. If no match is made, an error condition is returned to the Web server indicating access is denied and the Web server in turn sends the appropriate error back to the browser.

An important and unique point to make here is that document control server 12 must translate the URL prior to doing its search for a match on the URL. When an external business partner (user) enters the URL, they enter a URL where the first part of the URL points document control server 12 and the second portion is the 'role' associated with that URL.

e.g., https://www.<server ID>.com/Engineering/Standards/http__protocol.html

where

https=Secure http connection using SSL,

www.<server ID>.com=DNS name of document control server 12 (this name is unique to the customer installation)

/Engineering=role associated with this particular URL

/Standards/http__protocol.html=actual web page on the internal web server

If the real intranet web server associated with the Engineering role were engineer.abcd.com, then the translated URL that document control server 12 would search for is:

engineer.abcd.com/Standards/http__protocol.html

Note, the next two sections, Intranet connection and parsing the page, are entirely invisible to the end user.

Intranet Connection

If both the authentication and authorization phases completed successfully, document control server 12 will open a

TCP connection to the appropriate intranet server (engineer.abcd.com from the above example). Once the TCP connection has been made, document control server 12 generates an http request for the specific web page. The intranet server locates and returns the requested web page to document control server 12.

Parsing the Page

The pages returned by the intranet are categorized as either text or non-text. Examples of the latter are graphics, such as GIF or JPEG documents, sound objects, or executable objects, such as Java applets. Non-text pages are not parsed and forwarded back to the client browser unchanged. Text documents, such as HTML formatted pages, however, contain embedded links that may need to be translated into their external equivalent. Embedded links fall into 3 groups, some of which require translation, while others don't: relative path links, server path links and absolute path links.

Relative path links, which are of the form subdir/page.html, don't require translation because the browser will prepend the path based on the referrer's page. For example, if the referrer's page was at:

http://www.document_control_server.com/Engineering/Standards/http_protocol.htm

and the relative link was ssl_protocol.html, then the browser would prepend

http://www.document_control_server.com/Engineering/Standards/to the link.

Server path links take the form of /Specification/wheel.html and require translation. This type of link points to a page that resides on the same server as the referrer's page, but with an absolute path starting at the root directory of the server. Assuming the same referrer's page as in the paragraph above, the translated link would be /Engineering/Specification/wheel.html. (Note that the access string http:// is not required because the browser will fill that in.) The translation is performed by prepending the alias associated with the referrer's page, Engineering, in this case, to the path of the embedded link.

Absolute path links are full URLs, such as

http://engineer.abcd.com/Performance/testdrive.html

and require translation only if they point to a server that is in document control server 12's Alias table. The example link will get translated because it points to the server engineer.abcd.com that exists in document control server 12's Alias table as Engineering. The translation is done by replacing the intranet server's name by the document control server 12's server name, followed by the alias of the intranet server. In this example, the translated URL would be

http://www.<server 12 ID>.com/Engineering/Performance/testdrive.html.

Links that point to pages on servers unknown to document control server 12 are not translated because they may well point to valid external sites, such as Yahoo, which should be left untouched. In one embodiment, therefore, these links are not translated. (Note that if the referrer'page came in through the Secure Socket Layer (SSL), i.e., the URL starts with https://, then the translated links will also have https://.)

On the other hand, such links could pose a security threat. That is, the link could be pointing to an intranet server that contains sensitive information, whose existence should not be revealed to external users. To counter this, in one embodiment document control server 12 includes a list of links which should be hidden from the outside world. Links found on such a list would be translated to something innocuous.

Redirection

When a page has moved, an intranet server may send a redirect status back to document control server 12. This means that document control server 12 has to translate the redirected address, similar to how embedded links are handled, before forwarding it to the client browser.

Architecture

A document access system such as system 10 illustrated in FIG. 1 enables users to grant outside organizations direct access to internal web data in a secure, simple and manageable way. It is essentially a secure window through which outside partners can view internal web data. If, as is shown in FIG. 4, external interface 16 includes a firewall 40, system 10 also provides authenticated, authorized and view-customized business partner access to key Intranet servers via a standard web browser. Such a system enables users to easily, but accountably, grant authenticated partner access to internal web data, with complete control and authorization. Outside partners need only access a predefined URL in order to access an internal web page.

In one embodiment, as is shown in FIG. 4, document control server 12 is installed inside firewall 40. In another embodiment, as is shown in FIG. 5, document control server 12 is installed on a third network. Either way document control server 12 authenticates the outside user and then routes the request to a hidden, internal URL. The entire process is transparent to the outside user, and easily defined by the internal document control server 12 user. This allows business partners direct access to the data, eliminating the time lag, redundancy, lack of integrity and the bottleneck within the MIS Dept. (Please note that if document control server 12 is installed inside the firewall as is shown in FIG. 4, firewall 40 must be configured to restrict HTTP requests from any external source so that they can only get through to server 12. Similarly, if document control server 12 is installed on a third network (as a type of demilitarized zone (DMZ)) as is shown in FIG. 5, firewall 40 must be configured to restrict HTTP requests into internal network 20 so that they can only come from server 12.)

In a third embodiment, such as is shown in FIG. 6, document control server 12 is installed outside firewall 40 and is accessed through the Secure Sockets Layer (SSL). Such an embodiment should be set up so that firewall 40 allows HTTP traffic only from document control server 12 into internal network 20.

To further reduce any bottleneck, in one embodiment document control server 12 includes the option for the actual "data owners" themselves to define which partners have access to selective internal data. A data owner is a trusted individual within the organization that is empowered to grant Business Partners access privileges to Web pages on document servers 16. In one such embodiment, a Data Owner is assigned to one or more "roles," where a "role" represents the mapping alias assigned to one or the servers 16. A Data Owner can only add Business Partners or map URLs for the server "role" to which the Data Owner is assigned.

For example, an employee working in the Accounting department would be assigned to an Accounting role (server). The Accounting Data Owner is only able to access the internal servers specified by the administrator. This prevents the Accounting Data Owner from mapping URLs on any other server such as the Marketing or Engineering servers.

Once a Data Owner has been assigned to a role, he or she is able to perform the following tasks:

Add, modify, or delete a Business Partner from that particular role

Establish a user ID and password for a Business Partner for basic authentication

Post or map an internal URL for access by a Business Partner

Delete URLs from a posted Go List

Delegation of such tasks to the data owners frees up MIS while also delegating data administration to those who understand the information best. In such an embodiment, the system administrator also defines general authentication rules and the list of eligible document servers 16.

Business Partners are somewhat-trusted end users. They can be granted controlled access to selected Web page structures on internal Web servers(s) such as document servers 16 once they have been provided with the following information:

A URL connecting them to document control server 12.

A user ID and password to authenticate them to document control server 12.

The name of the "menu tag" that they will select when they connect to document control server 12 that will retrieve the internal Web pages as specified by the Data Owner.

It is the Data Owners' responsibility to create and maintain a listing of Business Partners that require access to the intranet servers they control and provide the Business Partner with the information they will need to access the selected intranet server(s).

Every Business Partner defined by a Data Owner is part of a "group." The "group" a Business Partner belongs to is directly related to the role a Data Owner has been assigned and what internal servers are associated with that role. These groups control what URLs they are able to access on the internal servers.

A Business Partner can be assigned to multiple groups. For example, a Business Partner may belong to both the Marketing and the Sales groups. Data Owners manage their Business Partner accounts through the Business Partner list. From the Business Partner List, a Data Owner can establish a new Business Partner and modify or delete an existing Business Partner to any groups that they control.

In one embodiment, a Business Partner List is accessed by clicking on a BP List button on a Data Owner Administration utility window.

In one embodiment, document control server 12 includes a go list processor 22 and a document processor 24 (see FIG. 3). Go list processor 22 determines if the user has authorization to access said first document. Document processor 24 reads a document from document server 14, cleans the document as detailed above and forwards a clean version of the document to the user. Go list processor 22 and document processor 24 will be discussed next.

a) The Go List

The Go list is used by the document control server 12 to determine which URLs an authenticated Business Partner may be allowed to display. The Go List is unique to each role. It is identified by the rolename.data within the roles/ directory. In one embodiment, the Go List is managed by MIS. Such an embodiment does not, however, take advantage of the flexibility provided by the architecture of the present invention. Instead, it can be advantageous to permit individual data owners to determine the URLs to be included in each Go List. Such an embodiment will be discussed next. In this example, documents are made available by the Data Owner and can be accessed by a user termed a "business partner (BP)".

In one such embodiment, the Go list contains data formatted as follows:

Real_url; MENU="menu_name"

where the Real URL is the actual URL (without the http://) used by document control server 12 to access that particular directory or file. The MENU parameter is always present. There can be a value within the quotes, or it can be empty. If there is a value within the quotes, then document control server 12 will parse that value up and set up a link to that particular URL with the title of the link being the Menu Name—if the Business Partner goes to its menu page after it logs in.

Only allowed URLs are present within the go list. No other URLs are included.

Also, the Go List will permit the Business Partner to access any of the files under a certain directory. For the time being, this is done by default on any URL that the user allows that ends with a "/"—to allow the Business Partner to access anything within the subdirectory—this is entered in the go list with the traditional "*" following the trailing slash. In one embodiment, the directory URL as kept intact and the Data Owner is given the option to cut and paste the path that the Data Owner wants the whole directory included in. In such an embodiment Data Owners append the * to the directory name if they want everything within that directory accessible to the Business Partners within that role. In another embodiment, explicit "disallows" could be included to handle documents the Data Owner wants to except from inclusion in the list of accessible documents.

b) The Mapping Code on Document Control Server 12

The next portion of the whole mapping design is where a lot of the real work comes into play. There is some code within document control server 12 that gets called when the user (Data Owner) wants to map a particular server to the Go List. In one embodiment, a graphical user interface is used to select URLs and business partners. In one such embodiment, this code gets activated by the Data Owner performing one of the following tasks:

(A) Bringing up the Server Go List for the first time

(B) Clicking on a Node within the Go List Mapping Tree that has not yet been expanded and may have some children node

(C) Clicking on the Remap Button

At this point the GUI communicates to Server 12 via a Get URL request. The Get URL request:

(A) indicates to the server to load the GO list for a particular role

(B) checks to see if the node has not already been expanded and that the node exists on this server (the front part of the URL indicating the server name is consistent) and that the node is of an html type (or directory type)—if the node matches all of these criteria, then the GUI will indicate to the server to expand the particular URL for the particular role and

(C) If the DO selects the Remap button, they are prompted to see if they want to remap that portion of the server down. If the DO selects yes, then a request to Remap with the currently selected URL and the role is sent to the server.

Server 12 then acts upon the request that it receives from the GUI

(A) if the request was to load the GO list, then the server portion checks for the existence of a role_map.data file in the roles directory. If this file does exist, then all that is done is that file is sent to the GUI line by line as is. If the file does not exist, then the file is created and the

mapping function is called. The mapping function is called with the file pointer, the name of the URL (the server name) to be mapped, and a depth indicator of 1. (NOTE: This embodiment includes an option to go down multiple layers, however due to timeout issues it may be better to just go down one layer at a time and let the user build the map as they see fit. If it goes down multiple layers than the mapping function must have the skills and capabilities to prevent the same URL from being mapped multiple times (the recursive nature of links and web spiders). The mapping code and its behavior is described below these ordered steps.)

Once the mapping code is done, then the URLs with their appropriate line syntax have been input and saved into the mapping file. The mapping file is then sent line by line to the GUI.

(B) If the request was to expand a node, the server code then calls the mapping function for the particular URL to be expanded. It opens up a temp file to be used to write the information into. It then calls the mapping code with a file pointer to this temp file, the URL to be mapped, and a depth of 1. (NOTE: same code called as in condition A, just different parameters). Once the mapping code returns, the file is communicated line by line to the GUI and then the file is removed from the system.

(C) If the request was to Remap a particular server or URL, then things get a little tricky. If the DO had chosen to remap the entire server, then the URL sent is the base URL—otherwise the URL sent was the URL that the DO wanted to remap from that point down. The same code is used regardless of the situation—just a different URL value. What happens is:

The current map file is copied over to the new map file until the line with the URL to be remapped is read in.

At this point this line is parsed to determine the depth in the tree (root level is 0).

A remap function is called which then does the following (note, that this is complicated by the fact that the tree could be at varying depths with files having been added or deleted and we would like to keep the prior shape and values of the tree when applicable)

Create a temp file to contain the intermediate results of the mapping

Call the map function with a pointer to the temp file, the URL, and a depth of 1.

It then compares the current map file line by line with the temp file.

If the URL at the current depth is not found in the respective depth in the new temp file, that URL and any URLs immediately following it with a depth greater than the current depth are removed (that initial file is missing)

If the URL at the current depth is found in the temp file, any URL lines in between the URL being searched for and the prior URL are new files and are added prior to the current URL in the map file. Their syntax lines indicate the current depth and default of disallowing that URL. Then the existing current URL line is left as is in the map file.

At this point the next URL in the map file needs to be examined, in examining this URL the URL line syntax is examined. The depth is the issue of primary concern. If the depth is the same as the current depth, then continue with this loop. If the depth is a depth deeper than the current depth, then this URL is a child of the prior URL and the prior URL then also needs to be remapped—the remap function is then called recursively with the prior URL. If the depth is less than the current depth, then we are no longer examining URLs that needed to be remapped and this function then returns.

After the final remap function returns, the remainder of the values are not to be touched and so they are copied over to the new map file as is.

After the server is done remapping, it then sends the whole newly mapped tree back to the GUI line by line. (NOTE: the server also then recreates the Go list to reflect the new values, information on creating the Go list from the mapping information is below.)

Finally, the GUI reads in the lines of information it gets from the server.

For (A) an initially loaded server, the GUI will create a tree examining each line of input. This is described in section 3 of this summary.

For (B) an expanded node, the GUI will create children nodes immediately below the expanded nodes by setting the depth according to the new tree and parsing each line of input. The parsing of the lines of input is described in section 3 of this summary.

For (C) a remapped server, the GUI will delete the previous tree and re-create the new tree (similar to A).

The code that does the mapping is a set of code pieces that loads the URL, parses any HTML that is returned, and creates a chain of information regarding the links. It then searches however deep is desired on each of the links.

it uses some standard libraries to aid in parsing and getting the URL and HTML

when it encounters a link, it stores that information in memory. At that time it also attempts to determine what sort of file it is—currently we only distinguish between the following: HTML, sound, graphic, external, video. It attempts to determine this based off of the hints surrounding it based on the filename and the surrounding html.

If the file is an HTML file and the mapping code has not yet reached the requested depth of mapping, the mapping code with then recursively try to bring up the HTML file and parse through its contents, etc.

As it comes across a file and finishes parsing it—it creates a syntax line that the GUI is expecting and writes this line to the file that it is passed. The line is as follows:

URL (just a tag) http://real_file _url (the is the URL to the file that the mapping code downloaded and parsed—this is the file that will be allowed or denied by the DO)

depth (an integer that is to indicate the current depth that this file is in the tree—the lines are listed such that the tree can be loaded via a depth—first sort of algorithm—it continues down the left hand side while the depth is getting bigger, adding children from left to right as the depth is the same, and going back up the tree as the depth becomes smaller)

filename (this is the name of the file for that URL—a * is used if the filename cannot be determined (like in the case of a directory or the server)

file type (this is a character which corresponds to the filetypes that were previously mentioned)

state of node (this is a character which indicates if this node was in a collapsed or expanded state when the tree

was saved—this is used by the GUI only, the mapping code always defaults this to C)

allowed (this is a character which indicates if this URL is to be allowed or disallowed, the mapping code defaults this to disallowed)

status of attaining the link (this is the HTTP status code of trying to access this link

it could have a value of 200 which means it was accessed OK, 404 meaning that this URL was not found, or a 0 indicating that this was not accessed)

already mapped (this is a one character flag used to indicate if this URL was already mapped and exists previously in the tree or not, this is most useful in a multi-depth mapping search)

Finally, the server has one more responsibility with respect to the mapping code. The server must handle writing out the saved data from the GUI when the DO requests to save the data. At this point, the data that is posted from the GUI to the server is written out line by line into the map file again. After this is done, the server deletes the prior Go List and parses through the map file a line at a time determining if that line is allowed. If the line is allowed the real_url part of the Map line is added (minus the http://) to the Go List. Next, the server checks the line to see if a "Menu" tag has been appended, if so, the server then adds the appropriate Menu tag to the Go list. Otherwise, it just adds a null menu tag to the Go list. As mentioned previously, currently if the real_url ends in a '/' an '*' is appended to the end of the real_url line to indicate that the user can access the entire directory—see Section 1 for further information.

After the Go List has been saved, the server is re-initialized with the new values—thus allowing immediate access or denial of access to the Business Partners for that role.

The Directory Map with the GUI

As mentioned previously, in one embodiment the GUI reads in and interprets the given line in order to creating nodes to represent the mapped server as a tree structure to the DO. Once such tree structure can be seen in FIG. 7.

The GUI communicates with server 12 as previously discussed and gets back well-defined lines of data. It parses the data and creates a node for each line provided by the Go list. It then looks at the expanded and collapsed feature to determine whether that node should be expanded or collapsed. It also looks at the file type to associate an image with that node. This image should allow the user to better determine what sort of nodes they are giving the Business Partners access to. The image is also determined by the return status from the mapping process—if a status of 404 is specified, then that link is determined to be broken (at least from document control server 12) and a broken link icon is displayed next to it. Finally, if that node is currently allowed, then the green ball is displayed next to it—to given the illusion of "green light means Go" to the DO.

The directory tree is created in a depth first fashion. It reads in each line examining the current depth. If the depth of the new node to be inserted is greater than the current depth than the node is inserted as a child of the previously inserted node. If the depth is the same, than it is inserted as a sibling. If the depth is less, then the tree is parsed back until the depth of the tree is at the same value of the depth and the node is inserted as a sibling at that level. The server is considered to be at the root level and thus has a depth of 0.

The text value that shows up with the node in the tree is the real_url value minus the http:// appended by the Menu Tag and Value if there is one for that node. A node may have a menu tag without being allowed.

The Data Owner (DO) can then traverse the tree examining the various links and determining what to allow and what to disallow. If they allow a directory—everything beneath that directory will be allowed. There currently is no mechanism for handling exceptions yet. If they allow files, then that file is allowed. If they have checked any other "include on allow" options (currently we offer, include all Gifs, Audio, Video, HTML, All Links)—then the files immediately beneath (once again only one layer deep) are automatically turned to allow if they are of the corresponding type. One note is that external files will never be "allowed"—as they do not exist on the server and thus it does not make sense for the DO to be allowing or disallowing those files.

When a data owner selects a link, they have the option of specifying the menu tag to be shown. If they do not specify one, then it is left blank. If they do then it is assigned for that node only. In order for it to get assigned to that node, the DO will have to select Allow or Disallow.

When Data Owners have finished making their changes they can save or cancel their mapping values. If they cancel then nothing that they did since their last save or remap will be saved. If they hit save, the values are communicated back to the server and the map file and the Go List File are updated as described previously.

Installation

Prior to install: Before fully installing and configuring document control server 12, the customer must have a digital certificate (for SSL encryption or transmissions) as well as set up and configure a server such as Internet Information Server (on NT) or Netscape Enterprise Server (on UNIX Solaris).

Installations: MIS installs document control server 12 onto the IIS or NES server and configures the firewall to allow HTTP access to the document control server 12 server.

Definition of end users: Document control server 12 offers organizations the option to delegate administration to end users who control the actual data (Data Owners) rather than forcing more work onto MIS. The data owner'access is defined by the network administrator. The data owner then maps which servers can be accessed. This data is stored in the document control server 12 "go list." The program code implementing document control server 12 is now completely installed, and ready for use.

Define business partner access: At this point, in order for an outside partner to access data, he/she must be granted access by the data owner. The data owner simply accesses the document control server 12 "data owner" GUI via a standard Java-enabled web browser. He/She can then define the new partner via role-based administration or explicitly choose which URLs may be accessed.

Outside users will not have access to any internal URL that is not specifically listed, even if there are embedded links in a URL for which access was granted. However, users can define access to a particular URL and all sub-pages as well.

Future partner access: After one role has been defined, future partners need only be added to that role, rather than requiring a whole new access to be defined.

Business partner access: All the outside partner has to do is type in the defined URL with any standard web browser. The partner will then be prompted for a user ID and password. Once these are entered, the partner will see a list of accessible URLs.

Back End Databases

It is important to understand that document control server 12 simply passes HTML information. This means document

control server 12 does not have a problem passing COI scripts and other dynamic content. Where this becomes particularly confusing is the access and authentication to back end databases via an application gateway.

One of document control server 12's greatest values is to allow outsiders access to ever-changing information such as order processing, shipping, etc. Much of this data is stored in large back-end databases with an application gateway on the front end. The application gateway puts an HTML front end on the database and allows Intranet users to query required information. Typically, a user only needs to enter a customer account number to access this information. However, in order to give outside users direct access, many organizations need to require some level of authentication to this process.

When document control server 12 passes an outside partner to any Intranet URL, the user is authenticated as a unique document control server 12 user, however, that user ID is not passed on to the Intranet server. Therefore, direct access to back end databases cannot be defined for each document control server 12 Business Partner. It will be necessary to create an HTML-based sign-in screen. This is a simple process and offers an opportunity for resellers and professional services to add value to the product sale.

Many Internet web servers handle restricted access in slightly different ways. If the user is not known, the web server responds with a 401 error. The user's browser then displays a standard screen requesting user ID and password. The user types these in and is granted access.

It is important to understand that document control server 12 cannot process this transaction. For security reasons, only HTTP traffic can pass through document control server 12. Any authentication must be HTTP based, as mentioned above.

In one embodiment, document control server 12 is installed on a standard web server running IIS (NT) or NES (Solaris). It requires no changes to the current infrastructure, and no "agents" or "clients" to be installed on any web servers or browsers.

Administration and data owner usage is accessed via document control server 12's Java user interface. This allows access via any web browser that supports Java (e.g., Internet Explorer 4.0, Netscape 4.0). Outside partner access is also accomplished via a standard web browser.

Operating with Third Party Firewalls

Document control server 12 can be used in conjunction with a firewall to add an additional layer of security to Business Partner communications via the Web. Two components need to be considered when determining the location of document control server 12: Domain Name Service (DNS) and routing.

Depending on the deployment option preferred and the capabilities of firewall 40, there are up to four different methods for routing traffic to, and through, server 12:

1) Redirected proxy—For added security on external to internal connections, a redirected proxy can be configured on your firewall to redirect the inbound connection requests. When a Business Partner on the external network attempts to connect to document control server 12, firewall 40 intercepts the request and establishes a connection to server 12. This rerouted connection hides the actual destination from the Business Partner requesting the connection.

2) Transparent proxy—A transparent proxy can be set up through firewall 40 to document control server 12. From the Business Partners' perspective it will appear as though they are connecting directly to server 12 and not connecting to the firewall first.

3) Directly to document control server 12—If document control server 12 is installed on the external side of firewall 40 (as in FIG. 6), connection requests will be routed directly to server 12. In such an embodiment, server 12 authenticates the Business Partner, and passes the request through firewall 40. Firewall 40 then retrieves the requested Web page(s) from the specified document server 16.

4) Through a third network—(some firewalls allow a "third network" capability, (sometimes called the DMZ or the Secure Server Network). The three deployment scenarios discussed above still apply in a "three network" environment, however, additional firewall configuration is necessary to ensure that the required name resolution (DNS), and routing are still possible.

Security Features SSL encryption. In one embodiment, data transmitted between the partner and web server is SSL-encrypted to prevent a sniffer from gathering information from the connection.

Document control server 12 server encrypted. Data stored on document control server 12 server such as user IDs, the go list, and partner profiles are all encrypted to prevent unauthorized access.

Password and user ID authentication. In one embodiment, document control server 12 supports password and user IDs for authorization. Stronger encryption could also be used.

Granular Access Controls

Business partners can only access internal URLs to which explicit access is given. If an accessible URL has embedded links to pages to which explicit access has not been granted, the partner cannot connect to them. However, if an embedded link is to an outside server, such as www.yahoo.com, in one embodiment access will not be restricted.

Internal URLs and IP Address Are Hidden

To ensure the security of the internal network and web pages, internal URLs and IP addresses are hidden from outside access. Partners type in a predefined URL and are presented with a list of accessible internal URLs. When a link is selected from the list Document control server 12 then maps to the internal URL. The internal URL and IP address are never displayed for the partner to see.

System Requirements, Compatibility and Performance

Considerations for performance and reliability include amount of cache memory, CPU power, BUS speed, amount of RAM, speed of memory chips, bus architecture (IDE, EIDE, PCI etc.), hard drive capacity, and hard drive quality (seek and access speeds). The following table identifies the basic characteristics of minimum, recommended and ideal server configurations to run document control server 12.

System Component	Minimum	Recommended	Ideal
CPU	Pentium 166	Pentium 200	Pentium Pro 200
RAM	32 MB	48 MB	64 MB
Hard disk	1 GB	2 GB	4 GB
Platform	IIS 3.0 (NT 4.0) or NES 3.0 (Solaris 2.5.1)		
Browser	Java enabled web browser MS Internet Explorer 4.0 or higher Netscape Navigator 4.0 or higher with Netscape's JDK 1.1 patch		
Other	CD-ROM, 3.5" diskette, Color monitor, Keyboard, Mouse		

Document control server 12 enables users to easily, but accountably, grant authenticated partner access to internal web data, with complete control and authorization. Outside partners need only access a predefined URL in order to access an internal web page.

The following examples provide a better idea of how document control server 12 can be used to meet a variety of needs.

Example—Manufacturing (Order Processing)

Manufacturing companies process thousands of orders every day. In order to keep up with competition and to achieve the highest levels of quality, customers/partners need to know the immediate status of an order to the minute. Many companies today have moved to just-in-time inventory systems to reduce overhead and costs. Document control server 12 can grant access directly to an order-processing page that connects directly into an order-processing database. The order-processing agents (data owners) can define what data customers/partners have direct access to. As a result, the customer knows immediately the status of an order. The supplier also saves money by eliminating the need to replicate data or take phone calls asking for updates.

Example—Distribution

A distribution environment operates an order-processing and shipping department very similar to manufacturing. However, distribution also requires various types of information to be distributed to different partners, such as pricing and quantity breaks. Document control server 12 allows a company to customize the view each distributor or reseller sees, such as pricing or quantity breaks.

Example—Financial Services

Financial institutions process millions of transactions a day with a large number of outside partners. These include the purchase and sale of assets as well as order/sale confirmation, etc. Today, many of these transactions require a third party to set up a secure certificate. Document control server 12 can speed up this entire process by allowing an agent to immediately allow an outside customer or partner access to trading information in minutes, and without the need for third-party intervention.

Example—Health Services

Health care and insurance organizations process thousands of claims each day. Partners need a secure way to pass medical information and process it into a company's systems. For example, a doctor treats a patient who has Blue Cross/Blue Shield. That doctor needs to know if the patient's insurance covers the treatment, then process the claim after the treatment is given, and finally check on the status of payment once a claim is submitted. With document control server 12, Blue Cross/Blue Shield can give the doctor's office access to their internal list of insured patients, as well as the status of current claims. The company no longer needs to replicate this data to a DMZ or SSN Internet server or handle a phone call. The doctor's office can also securely fill out a web-based claim form over the Internet to process the claim for treatment.

Example—Government Agency

It is necessary for various government agencies and departments to frequently share sensitive data. One example is the CIA and various law enforcement agencies. The FBI, DEA, ATF and other agencies must routinely check into the files of various personnel and public citizens. Typically, this requires these agencies to send a paper request for information to the CIA. The CIA must then search for the relevant information and then send a copy back to the requesting agency.

With document control server 12, the FBI and other agencies can be given direct access to the CIA files that might be relevant such as histories and fingerprint analysis databases. This can save time and money.

Document control server 12 offers several advantages over current methods such as cost savings, improved customer service and leveraging of the current infrastructure. Current methods for passing data to outside partners are expensive, slow and unreliable. Document control server 12 offers the information to partners faster, easier and cheaper. It also more tightly integrates partners, thus improving business relations. Document control server 12 also leverages the benefits of current technology such as the Internet and Intranet.

Other business advantages of document control server 12 include: it reduces overhead and costs; it eliminates the need to copy content to a web server within the DMZ or external network; it offers spontaneous, dynamic user-managed content; it eliminates the wait for an IS manager to update data or post on a web server; it eliminates integrity and replication issues; it more tightly integrates partners; and its open architecture allows access without the need to alter current technology.

Although specific embodiments have been illustrated and described herein, it will be appreciated by those of ordinary skill in the art that any arrangement which is calculated to achieve the same purpose may be substituted for the specific embodiment shown. This application is intended to cover any adaptations or variations of the present invention. Therefore, it is intended that this invention be limited only by the claims and the equivalents thereof.

What is claimed is:

1. A method of limiting access from an external network to documents stored on an internal network, the method comprising:

building a client list, wherein building a client list includes assigning each client to a role;

building a document list naming documents available to clients assigned to the client's role;

receiving a request for a document stored on the internal network;

associating the request with a client;

determining if the requested document is on the list of documents; and

if the requested document is on the list of documents, fetching the requested document as a proxy and sending the requested document to the client.

2. The method according to claim 1, wherein each document has a unique URL and wherein the document list includes the URL of each available document, wherein the step of determining includes the step of determining if the URL of the requested document is included in the document list.

3. The method according to claim 2, wherein building the document list includes displaying available documents in a tree structure within a graphical user interface, wherein displaying includes querying a document control server to obtain a current version of the document list.

4. The method according to claim 1, wherein associating the request with a client includes the step of authenticating the client.

5. The method according to claim 4, wherein each document has a unique URL and wherein the document list includes the URL of each available document, wherein the step of determining includes the step of determining if the URL of the requested document is included in the document list.

6. The method according to claim 5, wherein building the document list includes displaying available documents in a tree structure within a graphical user interface, wherein

displaying includes the step of querying a document control server to obtain a current version of the document list.

7. A document control system, including:

an internal network;

an external interface;

a document server connected to the internal network, wherein the document server controls access to a plurality of documents, including a first document; and

a document control server, wherein the document control server receives a document request for the first document, determines a user associated with the document request and authenticates the user, wherein the document control server includes a go list processor for determining if the user has authorization to access said first document and a document processor for reading the first document from the document server, cleaning the first document and forwarding a clean version of said first document to the user.

8. The document control system according to claim 7, wherein the external interface includes a firewall connected to an external network, wherein the document control server communicates to the document server through the firewall.

9. The document control system according to claim 7, wherein the document processor acts as a proxy to hide access to the first document.

10. The document control system according to claim 7, wherein the external interface includes a telephone interface into which a business partner can dial to gain access to the document control server.

11. A document control system, including:

an internal network;

an external interface;

a document server connected to the internal network, wherein the document server controls access to a plurality of documents, including a first document;

a document control server; and

a data owner interface for building a document list of available documents;

wherein the document control server receives a document request from the external interface for the first document, determines a user associated with the document request and authenticates the user; and

wherein the document control server includes a go list processor for determining, based on the document list, if the user has authorization to access said first document.

12. The document control system according to claim 11, wherein the data owner interface includes a graphical user interface which displays the document list in a tree structure, wherein the graphical user interface queries the document control server to obtain a current version of the document list.

13. The document control system according to claim 11, wherein the document control server further includes a document processor for reading the first document from the document server, cleaning the first document and forwarding a clean version of said first document to the user.

14. The document control system according to claim 13, wherein the document processor acts as a proxy to hide access to the first document.

15. The document control system according to claim 14, wherein the data owner interface includes a graphical user interface which displays the document list in a tree structure, wherein the graphical user interface queries the document control server to obtain a current version of the document list.

16. The document control system according to claim 11, wherein the external interface includes a firewall connected to an external network.

17. The document control system according to claim 11, wherein the external interface includes a telephone interface into which a business partner can dial to gain access to the document control server.

18. The document control system according to claim 8, wherein the document control server is connected to the external network and communicates to the firewall through the external network.

19. The document control system according to claim 8, wherein the document control server is connected to a third network and communicates to the firewall through the third network.

20. The document control system according to claim 7, wherein the document control server is connected to the internal network and wherein the external interface includes a firewall connected to an external network, wherein the firewall is configured to receive the document request and to route the document request to the document control server.

21. The document control system according to claim 7, wherein the document control server includes means for translating links embedded in the first document.

22. The document control system according to claim 7, wherein each user is assigned one or more roles and wherein the go list processor restricts access to documents as function of the role under which the user attempts to access the document.

23. In a system having an internal network and an interface to an external network, a method of handling requests from the external network for documents stored on the internal network, the method comprising:

defining one or more users;

defining documents accessible to the users;

receiving a document request from the external network;

determining a user associated with the document request;

authenticating the user associated with the document request;

determining if the user associated with the document request has permission to access the document requested; and

if the user associated with the document request has permission to access the document requested, retrieving the document requested from the internal network, cleaning the document of embedded links and delivering the document to the user associated with the document request.

24. The method according to claim 23, wherein each document has a unique URL and wherein determining if the user associated with the document request has permission to access the document requested includes:

accessing a document list listing the URL of each available document; and

generating an error message if the document requested is not on the document list.

25. The method according to claim 23, wherein defining documents accessible to the users includes assigning each

user to one or more roles and limiting access to documents as a function of role and wherein determining if the user associated with the document request has permission to access the document requested includes determining if users in the role associated with the document request have permission to access the document requested.

26. The method according to claim 24, wherein defining documents accessible to the users includes assigning each user to one or more roles and limiting access to documents as a function of role and wherein determining if the user associated with the document request has permission to access the document requested includes determining if users in the role associated with the document request have permission to access the document requested.

27. The method according to claim 23, wherein each document request includes an HTTP header and wherein authenticating the user associated with the document request includes retrieving authentication information from the HTTP header.

28. The method according to claim 23, wherein cleaning the document of embedded links includes looking for a server path link and replacing the server path link with a link to an alias.

29. The method according to claim 23, wherein cleaning the document of embedded links includes looking for an absolute path link, determining if the absolute path link is a link which should be hidden and, if the absolute path link is a link which should be hidden, replacing the absolute path link with a different link.

30. In a system having an internal network and an interface to an external network, a method of handling requests from the external network for documents stored on the internal network, the method comprising:

defining a plurality of users, including a first and a second user;

assigning each user to one or more roles, wherein assigning includes assigning the first user to a first role and the second user to a second role;

defining documents accessible to the users, wherein defining includes limiting access to documents as a function of the roles assigned to the user;

receiving a document request from the external network;

determining a user and a role associated with the document request;

authenticating the user associated with the document request;

determining if users in the role associated with the document request have permission to access the document requested; and

if users in the role associated with the document request have permission to access the document requested, retrieving the document requested from the internal network and delivering the document to the user associated with the document request.

31. The method according to claim 30, wherein each document has a unique URL and wherein determining if the user associated with the document request has permission to access the document requested includes:

accessing a document list listing the URL of each available document; and

generating an error message if the document requested is not on the document list.

32. The method according to claim 30, wherein retrieving the document requested includes cleaning the document of embedded links.

33. The method according to claim 32, wherein cleaning the document of embedded links includes looking for a server path link and replacing the server path link with a link to an alias.

34. The method according to claim 32, wherein cleaning the document of embedded links includes looking for an absolute path link, determining if the absolute path link is a link which should be hidden and, if the absolute path link is a link which should be hidden, replacing the absolute path link with a different link.

35. A computer-readable medium having program code for limiting access from an external network to documents stored on an internal network, the program code comprising:

program code for building a client list, wherein program code for building a client list includes program code for assigning each client to a role;

program code for building a document list naming documents available to clients assigned to the client'role;

program code for receiving a request for a document stored on the internal network;

program code for associating the request with a client;

program code for determining if the requested document is on the list of documents; and

program code for, if the requested document is on the list of documents, fetching the requested document as a proxy and sending the requested document to the client.

36. A computer-readable medium comprising program code, in a system having an internal network and an interface to an external network, for handling requests from the external network for documents stored on the internal network, the program code comprising:

program code for defining one or more users;

program code for defining documents accessible to the users;

program code for receiving a document request from the external network;

program code for determining a user associated with the document request;

program code for authenticating the user associated with the document request;

program code for determining if the user associated with the document request has permission to access the document requested; and

program code for, if the user associated with the document request has permission to access the document requested, retrieving the document requested from the internal network, cleaning the document of embedded links and delivering the document to the user associated with the document request.

37. A computer-readable medium comprising program code, in a system having an internal network and an interface to an external network, for handling requests from the external network for documents stored on the internal network, the program code comprising:

program code for defining a plurality of users, including a first and a second user;

program code for assigning each user to one or more roles, wherein assigning includes assigning the first user to a first role and the second user to a second role;

program code for defining documents accessible to the users, wherein defining includes limiting access to documents as a function of the roles assigned to the user;

program code for receiving a document request from the external network;

program code for determining a user and a role associated with the document request;

program code for authenticating the user associated with the document request;

program code for determining if users in the role associated with the document request have permission to access the document requested; and

program code for, if users in the role associated with the document request have permission to access the document requested, retrieving the document requested from the internal network and delivering the document to the user associated with the document request.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 6,357,010 B1
DATED : March 12, 2002
INVENTOR(S) : Viets et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Title page,
Item [54], delete "**SYSTEM AND METHOD FOR CONTROLLING ACCESS TO DOCUMENTS STORED ON AN INTERNAL NETWORK**" and insert -- **ACCESS CONTROL TO INTERNAL NETWORK DOCUMENTS WITH CLIENT LIST ASSIGNING CLIENT'S ROLE AND DOCUMENT LIST NAMING DOCUMENTS AVAILABLE TO CLIENTS ASSIGNED TO CLIENT'S ROLE** --, therefor.

Column 1,
Line 48, delete "company'firewall" and insert -- company's firewall --, therefor.

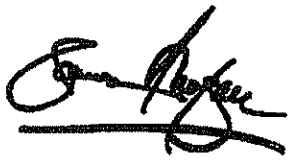
Column 2,
Line 7, delete "client'role" and insert -- client's role --, therefor.

Column 4,
Line 28, delete "roles" and insert -- role --, therefor.

Column 20,
Line 23, delete "client'role" and insert -- client's role --, therefor.

Signed and Sealed this

Twenty-fifth Day of March, 2003



JAMES E. ROGAN
Director of the United States Patent and Trademark Office

US007185361B1

(12) **United States Patent**
Ashoff et al.

(10) **Patent No.: US 7,185,361 B1**
(45) **Date of Patent: Feb. 27, 2007**

(54) **SYSTEM, METHOD AND COMPUTER PROGRAM PRODUCT FOR AUTHENTICATING USERS USING A LIGHTWEIGHT DIRECTORY ACCESS PROTOCOL (LDAP) DIRECTORY SERVER**

(75) Inventors: **Thomas D. Ashoff**, Mt. Airy, MD (US); **Steve O. Chew**, Pittsburgh, PA (US); **Jeffrey J. Graham**, Olney, MD (US); **Andrew J. Mullican**, Columbia, MD (US)

(73) Assignee: **Secure Computing Corporation**, St. Paul, MN (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/495,157**

(22) Filed: **Jan. 31, 2000**

(51) **Int. Cl.**
H02H 3/05 (2006.01)

(52) **U.S. Cl.** **726/4**; 713/151; 713/154; 707/1; 726/2; 726/8; 726/11; 726/12; 726/13; 726/14

(58) **Field of Classification Search** 713/201, 713/151–154; 707/1; 726/4, 8, 11–14
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,657,390	A *	8/1997	Elgamal et al.	713/151
5,898,830	A *	4/1999	Wesinger, Jr. et al.	713/201
6,047,322	A *	4/2000	Vaid et al.	709/224
6,131,120	A *	10/2000	Reid	709/225
6,182,142	B1 *	1/2001	Win et al.	709/229
6,212,558	B1 *	4/2001	Antur et al.	709/221
6,233,688	B1 *	5/2001	Montenegro	713/201
6,324,648	B1 *	11/2001	Grantges, Jr.	713/201
2003/0126468	A1 *	7/2003	Markham	713/201

OTHER PUBLICATIONS

Microsoft Corporation. Microsoft Computer Dictionary, Microsoft Press, Third edition, p. 197.*
Definition of application gateway, Webopedia computer dictionary, http://www.webopedia.com/TERM/A/application_gateway.html.*
Definition of firewall, Webopedia computer dictionary, http://www.webopedia.com/TERM/firewall.html.*
Netegrity, SiteMinder 3.5 Architecture.
How to Securely Manage and Control User Access to E-Commerce Web Sites, Netegrity White Paper, Jul. 1999.
Check Point Account Management Client, Version 1.0, Sep. 1998.
FireWall-1 Architecture and Administration, Chapter 4, pp. 135-154, Sep. 1998.
Howes et al., *The LDAP Application Program Interface*, University of Michigan, Aug. 1995.

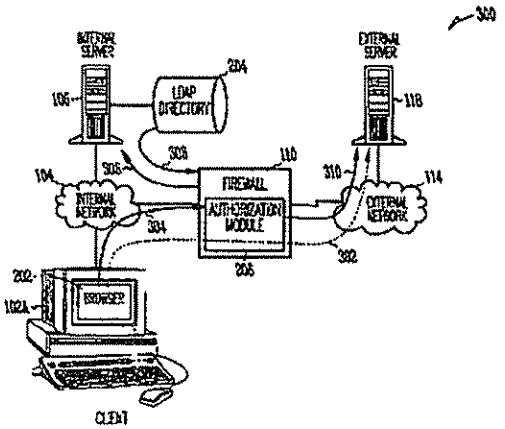
* cited by examiner

Primary Examiner—Taghi T. Arani
(74) *Attorney, Agent, or Firm*—Schwegman, Lundberg, Woessner & Kluth, P.A.

(57) **ABSTRACT**

A system, method and computer program product for providing authentication to a firewall using a lightweight directory access protocol (LDAP) directory server is disclosed. The firewall can be configured through a graphical user interface to implement an authentication scheme. The authentication scheme is based upon a determination of whether at least part of one or more LDAP entries satisfy an authorization filter.

15 Claims, 5 Drawing Sheets



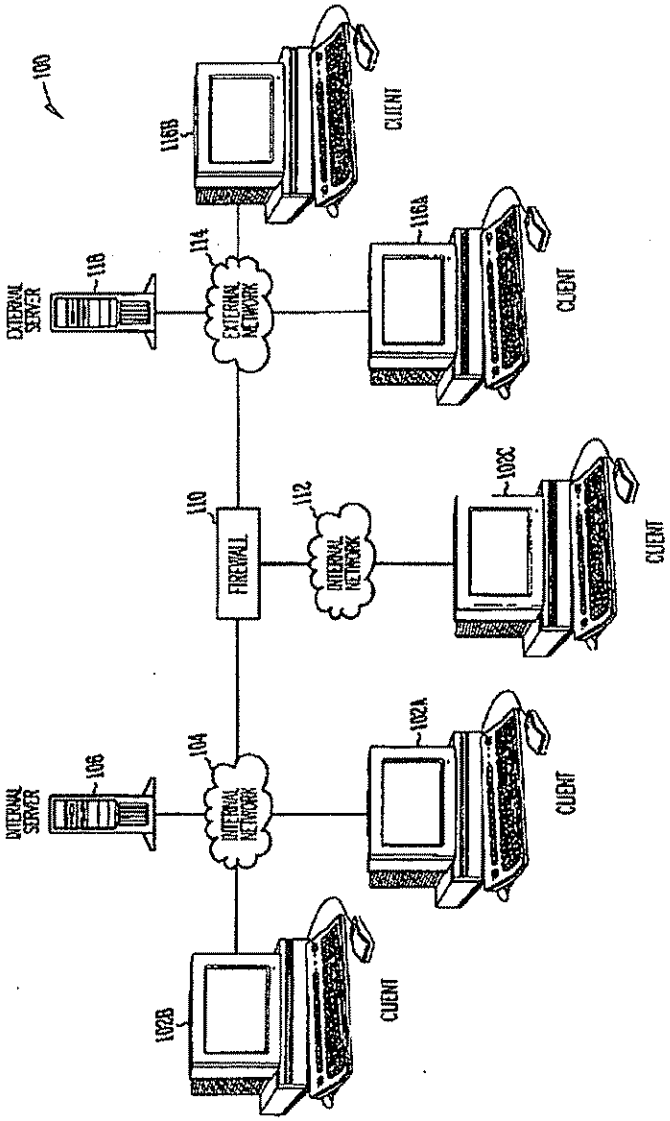


Fig. 1 (Prior Art)

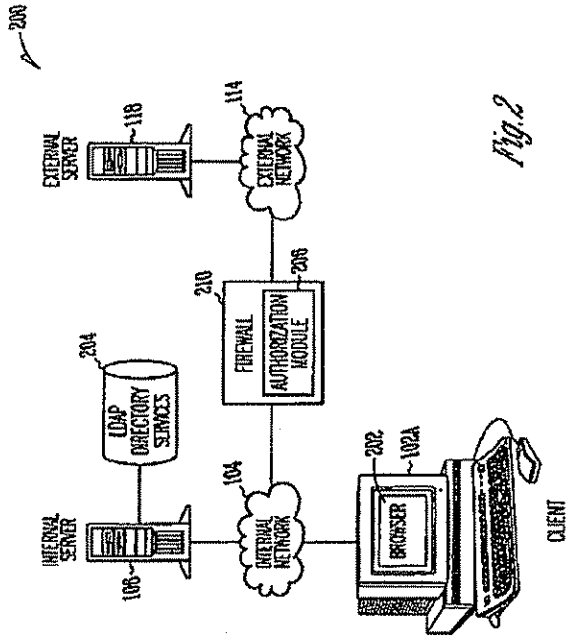

200
EXTERNAL SERVER
118
114
EXTERNAL NETWORK
210
FIREWALL
AUTHORIZATION MODULE
206
204
LDAP DIRECTORY SERVICES
INTERNAL SERVER
106
104
INTERNAL NETWORK
202
102A
BROWSER
CLIENT

Fig. 2

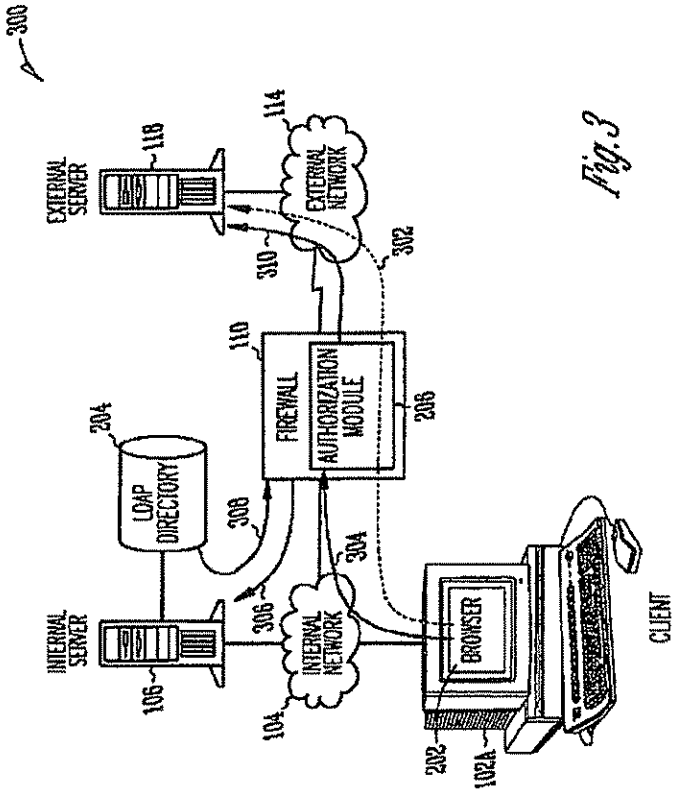

300
EXTERNAL SERVER
118
EXTERNAL NETWORK
114
310
302
110
FIREWALL
AUTHORIZATION MODULE
206
204
LDAP DIRECTORY
308
304
INTERNAL SERVER
106
306
INTERNAL NETWORK
104
BROWSER
202
102A
CLIENT

Fig. 3

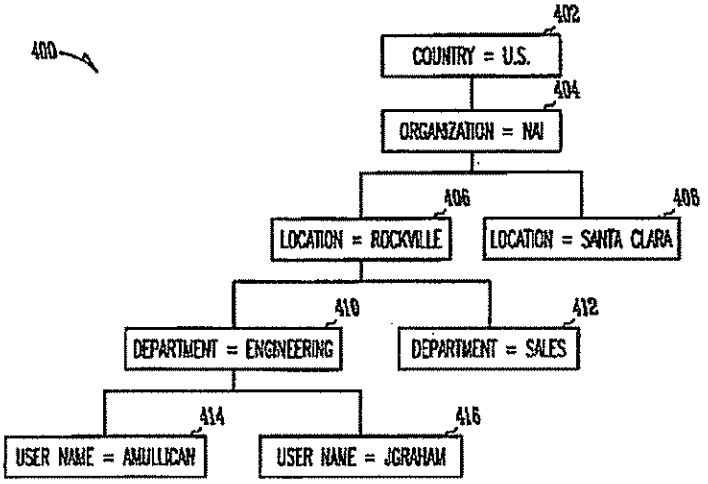

400
402
COUNTRY = U.S.
404
ORGANIZATION = NAI
406
LOCATION = ROCKVILLE
408
LOCATION = SANTA CLARA
410
DEPARTMENT = ENGINEERING
412
DEPARTMENT = SALES
414
USER NAME = AMULLICAN
416
USER NAME = JGRAHAM

Fig. 4

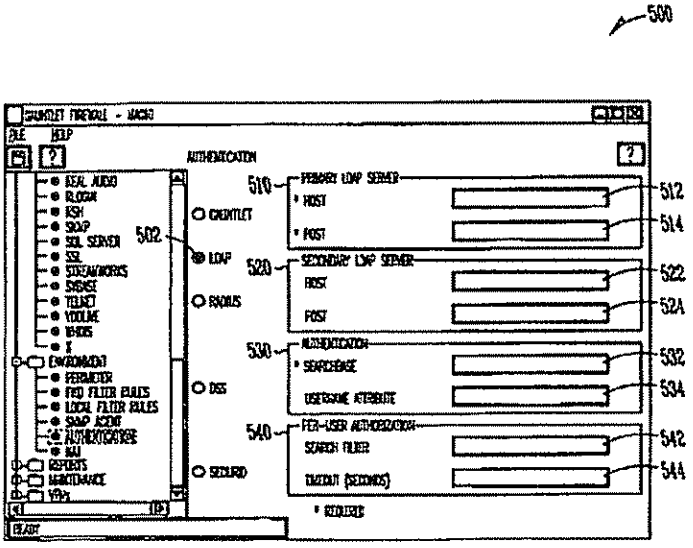

500
502
510
512
514
520
522
524
530
532
534
540
542
544
AUTHENTICATION
GAUNTLET
LDAP
RADIUS
DSS
SECURID
HOST
PORT
SEARCHBASE
SEARCH FILTER

Fig.5

SYSTEM, METHOD AND COMPUTER PROGRAM PRODUCT FOR AUTHENTICATING USERS USING A LIGHTWEIGHT DIRECTORY ACCESS PROTOCOL (LDAP) DIRECTORY SERVER

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to user authentication mechanisms and more particularly to user authentication mechanisms for firewalls.

2. Related Art

Control over access to information technology (IT) resources is a common need today. A firewall can be used to protect IT resources behind the firewall. Network firewalls can enforce a site's security policy by controlling the flow of traffic between two or more networks. For example, a company might encourage file transfers to the company's network that assist employees, but might discourage file transfers of potentially sensitive company confidential information from the company network to external destinations. Firewalls often are placed between a corporate network and an external network such as, e.g., the Internet, or a partnering company's network. Firewalls can also be used to segment parts of a corporate network. A firewall system can provide both a perimeter defense to, e.g., an internal network, and a control point for monitoring access to and from specific networks such as, e.g., an external network.

Firewalls can control access at a network level, an application level, or both. At the network level, a firewall can restrict packet flow based on protocol attributes. For example, the packet's source address, destination address, originating transmission control protocol/user datagram protocol (TCP/UDP) port, destination port, and protocol type can be used for the control decisions. At an application level, a firewall can participate in communications between the source and destination applications with the firewall's control decisions being based on details of the conversation and other available information such as, e.g., previous connectivity or user identification. Thus, a firewall can authenticate users to control access to and from IT resources behind and before the firewall.

Firewalls can be packaged as system software, combined hardware and software, and, more recently, dedicated hardware appliances (e.g., embedded in routers, or easy-to-configure integrated hardware and software packages that can run on dedicated platforms). An example of an application-based firewall is the Gauntlet™ firewall available from Network Associates, Inc.

Firewalls can defend against attacks ranging from, e.g., unauthorized access, IP address "spoofing" (i.e., a technique by which hackers disguise traffic as coming from a trusted address to gain access to a protected network or resource), buffer overrun attacks, session hijacking, viruses and rogue applets, and rerouting of traffic. However, inherent limitations exist in certain services and protocols that conventional firewalls cannot remedy.

Conventionally, when software application programs sought to restrict what a user could do with the programs, the programs required identification of the user. For example, if a user desires access to sensitive corporate financial data in an accounting program, access to the data can be restricted by means of authentication mechanisms such as, e.g., a password. The application program therefore requires a list of users and identification information for the user for use in authenticating the user.

Early software application programs often included their own integrated authentication mechanisms. Users often use a variety of software application programs, each possibly having its own authentication mechanism. Users find it cumbersome to remember different passwords associated with each of the multiple software application programs.

IT resources used by companies today can include access to multiple software application programs and Internet based applications. For example, employees at a given company can use e-mail and groupware applications, and other office automation programs including, e.g., to spreadsheets, word-processors and presentation programs. As every application program conventionally has its own authentication mechanism, a separate database is initialized and updated for each application.

Authentication mechanisms can use a query to a database known as a directory that can store information about users. A directory is similar to a database in that one can store information in a directory and later retrieve the information from it. However, a directory is specialized in that a directory is typically designed for reading more than writing. A directory offers a static view of the information and allows simple updates without transactions. Thus, while a database is typically written to and read from frequently, a directory by comparison is primarily read from and is infrequently updated.

A directory service includes all the functions of a directory and adds a network protocol that can be used to access the directory. Standardization is desirable in implementing a directory service.

An early standard for directory service was the directory access protocol (DAP), which originated in the European standards organization. DAP although specifying a vast, feature-rich protocol for storing and encoding directory information, was unwieldy in size.

Today, a new protocol, lightweight directory access protocol (LDAP), is gaining wide acceptance in business. The LDAP standard defines an information model for a directory, a namespace for defining how directory information is referenced and organized, and a network protocol for accessing information in the directory. LDAP can also include an application programming interface (API). The LDAP protocol mandates how client and server computers can communicate with a LDAP directory. However, LDAP does not mandate how data should be stored. More and more companies today use an LDAP directory server to store a database of employees. The LDAP directory generally can store an employee name, phone number, address and other information about the employee, and a password for modifying the employee's information.

Firewalls also maintain a database of users and are operative to prompt users for an identifying user identifier and password. These conventional firewalls require that employee names and passwords be entered into a firewall authentication database. Maintenance of the firewall authentication database is especially burdensome where there are a large number of employees that are frequently leaving or joining a company or when a company has a large number of firewalls. Accordingly, what is needed is a mechanism for reducing this administrative burden. More specifically, what is needed is a mechanism for leveraging an existing LDAP directory server as part of a firewall's authentication process. In this manner, an existing LDAP directory server can be used as a central directory that stores the data used by all applications.

SUMMARY OF THE INVENTION

A system, method and computer program product for enabling the authentication of users to a firewall using a lightweight directory access protocol (LDAP) directory server is provided by the present invention. The firewall can be configured through a graphical user interface to implement an authentication scheme. The authentication scheme is based upon a determination of whether information contained in one or more LDAP entries satisfy an authorization filter. It is a feature of the present invention that the authentication scheme can be configured independently of specifically stated field requirements or schema of the firewall. In accordance with the present invention, the authentication scheme can be flexibly specified to interact with a LDAP directory that has been uniquely developed for a company's internal needs. The company's investment in its existing administrative infrastructure can therefore be leveraged to a greater degree.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other features and advantages of the invention will be apparent from the following, more particular description of a preferred embodiment of the invention, as illustrated in the accompanying drawings.

FIG. 1 illustrates a communications network including a firewall.

FIG. 2 illustrates a communications network including a lightweight directory access protocol (LDAP) directory server and an authorization module within a firewall.

FIG. 3 illustrates the authentication of a client user through a firewall.

FIG. 4 illustrates an example embodiment of an LDAP directory tree.

FIG. 5 illustrates an embodiment of a graphical user interface for configuring the LDAP authentication feature.

DETAILED DESCRIPTION OF THE INVENTION

A preferred embodiment of the invention is discussed in detail below. While specific implementations are discussed, it should be understood that this is done for illustration purposes only. A person skilled in the relevant art will recognize that other components and configurations may be used without parting from the spirit and scope of the invention.

FIG. 1 illustrates an example embodiment of a communications network 100 including client computers 102*a* and 102*b* coupled via an internal network 104 to an internal server computer 106 and a firewall 110. Communications network 100 also includes a client computer 102*c* coupled via an internal network 112 to firewall 110. Finally, communications network 100 includes client computers 116*a* and 116*b* coupled via an external network 114 to an external server computer 118 and firewall 110. External network 114 can represent, e.g., the Global Internet, or a partnering company's network.

Network firewall 110 can enforce a business' security policy by controlling the flow of traffic between two or more networks such as, e.g., internal networks 104 and 112 and external network 114. In general, firewall 110 serves to isolate internal networks 104 and 112 from one another and also from external network 114.

As illustrated in FIG. 1, firewall 110 can be used to segment parts of a corporate network. For example, firewall 110 can be used to control information flow between a corporation's internal networks 104, 112. Firewall 110 can also provide a perimeter defense between an internal network 104, 112 and an external network 114.

FIG. 2 illustrates an example embodiment of a communications network 200 that includes client computer 102*a* coupled via internal network 104 to internal server 106 and to firewall 210. Firewall 210 is also coupled via external network 114 to external server 118.

As shown, client computer 102*a* includes a browser 202. Browser 202 can in one embodiment be an Internet browser that provides a graphical user interface to network resources. Browser 202 is generally operative to parse and make requests to network resources such as, e.g., external server 118, and present the results of the request to a client user viewing client computer 102*a*.

Internal server 106 is shown including a lightweight directory access protocol (LDAP) directory 204, which can be configured to store employee information. For example, a human resources database could be stored as an LDAP directory having a directory structure such as that illustrated in FIG. 4. As illustrated, LDAP directory tree 400 includes country 402 set in this example to US, organization 404 set to NAI, location 406 set to Rockville and location 408 set to Santa Clara, department 410 set to engineering and department 412 set to sales, and username 414 set to smallican and username 416 set to jgraham.

External server 118 can include an Internet server application. In one embodiment, the Internet server application supports file transfer protocol (FTP) communication. As would be apparent to those skilled in the relevant art, other types of server applications can be included on external server 118 including, e.g., databases, and electronic mail.

Firewall 210 is shown including an authorization module 206. Authorization module 206 is used to authenticate a client user (e.g., client computer 102*a*) to determine if the client user's communication is authorized to pass through firewall 210. Conventional firewalls 110 included their own database having a list of users and passwords, to enable authentication through firewall 110.

In accordance with the present invention, firewall 210 does not authenticate users using its own database. Rather, firewall 210 authenticates users using information contained within LDAP directory 204. As will be described in greater detail below, firewall 210 can authenticate users through an authentication scheme that can be based upon the unique composition of an organization's LDAP directory 204.

It is a feature of the present invention that the authentication scheme of the present invention can operate independently of specifically stated field requirements or schema of the firewall 210. In other words, an organization's LDAP directory 204 need not be modified to conform to a schema imposed by the firewall 210. Moreover, resistance to such a modification will not result in the maintenance of multiple directories.

In accordance with the present invention, the authentication scheme can be flexibly specified to interact with an existing LDAP directory that has been uniquely developed for a organization's internal needs. This framework enables a firewall administrator to seamlessly integrate a firewall product into an existing administrative infrastructure. The organization's investment in the existing administrative infrastructure can therefore be leveraged to a greater degree.

FIG. 3 illustrates the authentication process that is implemented by firewall 210. In the illustrated example, firewall 210 authenticates a client user at client computer 102*a* running a browser 202 that is attempting to access an

application or resource on external server 118. This access path is illustrated by path 302.

This authentication process begins when client computer 102a initiates a network resource request 304 from browser 202. The network resource request 304 is intercepted by firewall 210. Authorization module 206 within firewall 210 challenges the client user to identify himself or herself. A challenge could in one embodiment include a request for entry of a username and password. Upon receipt of the identification information, authorization module 206 searches an authentication database (not shown) to identify an authentication method (e.g., LDAP authentication). If no entry in the authentication database is found for the client user, then a default authentication method can be used. In the LDAP authentication process, authorization module 206 binds to LDAP directory 204 and uses the userPassword attribute for authentication.

After authorization module 206 authenticates the client user, authorization module 206 then determines whether the client user is authorized to have his access request fulfilled. The LDAP authorization process is illustrated as communications 306 and 308. Communications 306 and 308 are facilitated using the LDAP protocol and may utilize the secure sockets layer.

If per-user authorization is configured, authorization module 206 determines whether one or more attributes of the client user's LDAP entry satisfies an authorization filter. If the one or more attributes of the client user's LDAP entry does not satisfy the authorization filter, then authorization module 206 determines that the authorization fails. If the authorization filter is satisfied, then the client user's network resource request is allowed through firewall 210. This allowed connection is illustrated in FIG. 3 as path 310.

To support per-user authorization, an administrator configures an authorization filter to use when authenticating users. One or more attributes in the client user's LDAP directory entry and associated values can be selected for the authorization filter. Once configured, authorization module 206 can verify that the LDAP entry used in the bind call satisfies the authorization filter before allowing the user access to/through the firewall.

FIG. 5 illustrates an example embodiment of a graphical user interface (GUI) 500 of a firewall systems administrator application screen. As shown by a selected radio button, LDAP authentication 502 has been selected. GUI 500 includes a primary LDAP server settings area 510, a secondary LDAP server settings area 520, an authentication settings area 530, and a per-user authorization settings area 540.

The primary LDAP server settings area 510 includes a host field 512 and a port field 514. The host field 512 can be used to enter an IP address or host name of a primary LDAP server. The port field 514 can be used to enter the port to be used on the primary LDAP server.

The secondary LDAP server settings area 520 also includes a host field 522 and a port field 524. The host field 522 can be used to enter an IP address or host name of a secondary LDAP server. The port field 524 is used to enter the port to be used on the secondary LDAP server. Fields 522, 524 can be left blank if no secondary LDAP server is being used.

The authentication settings area 530, can include searchbase field 532 and a username attribute field 534. The searchbase field 532 can be used to indicate the top of the directory tree 400 such as, e.g., country 402, organization 404, location 406, and department 410, so that a lookup can be within that portion of the directory tree. For example, a set of attribute pairs such as, e.g., o=NAI, c=US to append to all requests to the LDAP server can be entered. The username attribute field 534 can include a default username attribute such as, e.g., uid. The username attribute field 534 can be used in performing per-user authorization.

The per-user authorization settings area 540 includes a search filter field 542 and a timeout field 544. The timeout field 544 can include a default value such as, e.g., 60 seconds. For example, timeout field 544 can be used to limit the amount of time the authorization filter query can take. If the time is exceeded, the authorization fails.

The search filter field 542 is used by firewall 210 in identifying the appropriate fields that are the subject of the LDAP directory authentication query. Upon receipt of a response from the LDAP directory 204, firewall 210 can then determine whether the client user is authorized to authenticate through the firewall 210.

In general, the authorization filter can contain any LDAP-valid combination of attributes and values, including object classes. At its simplest, the authorization filter specifies a single attribute and value pair. For example, the search filter field 542 can be used to enter a search filter expression such as "objectclass=gauntletUser."

Consider another example where LDAP directory 204 is configured by the company to include a field that would provide an access code level for each user. For example a "1" could correspond to only e-mail access, while a 5 could mean full access to all Internet services including world wide web browsing. In this environment, an authorization filter can be specified as "(&(objectclass=gUser)(status>=5))".

It should be noted that the authorization process need not be based on per-user authorization. In another embodiment, the authorization process can be based on a per-service authorization. In this embodiment, the per-service authorization can include an authorization for protocol services. Examples of protocol services include FTP, simple mail transport protocol (SMTP) e-mail, hypertext transport protocol (HTTP), etc. The per-service authorization can also be based on LDAP directory information. For example, authorization module 206 can use group memberships to determine whether a client user can use HTTP through firewall 210. To satisfy this authorization process, the authenticated user must be a member of the "web-users" group in the LDAP directory.

In one embodiment, the per-service authorization process uses the standard groupOfNames and groupOfUniqueNames object classes for authorization decisions. In general, a mechanism can be included that supports the specification of arbitrary group names for each service to be controlled. Control can then be based on a per-proxy basis or a per-policy basis.

Specification of per-service authorization criteria can also be implemented using the search filter field 542. In general, a different search (or authorization) filter can be provided for each service. For example, a search filter field can be included in GUI 500 to determine whether, e.g., a user is authorized to perform a file transfer, to send e-mail, or to access the world wide web. A search filter field can also be included in GUI 500 to determine whether, e.g., a user is a member of a particular group such as, e.g., engineering department 410, and if so, then particular services can be authorized based on being part of that group.

As noted, it is a feature of the present invention that firewall 210 can support arbitrary LDAP directory schema. Accordingly, firewall 210 does not require additional firewall-specific object classes or attributes in the directory.

Customers can populate the LDAP directories with whatever data they require. This authentication environment can be flexibly applied across multiple organizations each having their own sets of directory information. Indeed, the concepts of the present invention can be used to implement an authorization filter that relies on portions of information that are stored in distinct LDAP directories. This distributed authentication scheme enables an organization to implement segmented management of the user database.

While various embodiments of the present invention have been described above, it should be understood that they have been presented by way of example only, and not limitation. Thus, the breadth and scope of the present invention should not be limited by any of the above-described exemplary embodiments, but should be defined only in accordance with the following claims and their equivalents.

What is claimed is:

1. A system for authorizing client access to a network resource, comprising:

a server having at least one directory that can be accessed using a network protocol, said at least one directory being configured to store information concerning an entity's organization; and

a firewall that is configured to intercept network resource requests from a plurality of client users on an internal network, said firewall being operative to authorize a network resource request based upon a comparison of the contents of at least part of one or more entries in said at least one directory to an authorization filter, wherein said authorization filter is generated based on a directory schema that is predefined by said entity.

2. The system of claim 1, wherein said at least one directory is a lightweight directory access protocol directory.

3. The system of claim 1, wherein said authorization filter is specified using a graphical user interface.

4. The system of claim 1, wherein said authorization filter implements a per-user authentication scheme.

5. The system of claim 1, wherein said authorization filter implements a per-service authentication scheme.

6. The system of claim 1, wherein said firewall and said directory communicate using secure socket layer communication.

7. The system of claim 1, wherein said firewall is configured to query multiple directories.

8. An authentication method at a firewall, comprising the steps of:

(a) receiving a network resource request from a client user at an internal network;

(b) querying, using a network protocol, at least one directory that is configured to store information concerning an entity's organization, wherein said query is based upon an authorization filter that is generated based on a directory schema that is predefined by said entity;

(c) determining, based on the results of said query, whether the contents of at least part of one or more entries in said at least one directory satisfy said authorization filter; and

(d) permitting said network resource request through said firewall if said authorization filter is satisfied.

9. The method of claim 8, wherein step (b) comprises the step of querying said at least one directory using a lightweight directory access protocol.

10. The method of claim 8, further comprising the step of specifying an authorization filter using a graphical user interface.

11. The method of claim 10, wherein said specifying step comprises the step of specifying an authorization filter that implements a per-user authentication scheme.

12. The method of claim 10, wherein said specifying step comprises the step of specifying an authorization filter that implements a per-service authentication scheme.

13. The method of claim 8, wherein step (b) comprises the step of querying said directory using secure socket layer communication.

14. The method of claim 8, wherein step (b) comprises the step of querying multiple directories.

15. A computer program product for enabling a processor in a computer system to implement an authentication process, said computer program product comprising:

a computer usable medium having computer readable program code embodied in said medium for causing a program to execute on the computer system, said computer readable program code comprising:

first computer readable program code for enabling the computer system to receive a network resource request from a client user at an internal network;

second computer readable program code for enabling the computer system to query, using a network protocol, at least one directory that is configured to store information concerning an entity's organization, wherein said query is based upon an authorization filter that is generated based on a directory schema that is predefined by said entity;

third computer readable program code for enabling the computer system to determine, based on the results of said query, whether the contents of at least part of one or more entries in said at least one directory satisfy said authorization filter; and

fourth computer readable program code for enabling the computer system to permit said network resource request through a firewall if said authorization filter is satisfied.

* * * * *

UNITED STATES DEPARTMENT OF COMMERCE
Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.
08/964,388	11/06/97	TOUBOUL	S 40492.H001C

LM02/0107

GRAHAM & JAMES
600 HANSEN WAY
PALO ALTO CA 94304-1043

EXAMINER	
SHAW, B	
ART UNIT	PAPER NUMBER
2786	

DATE MAILED: 01/07/99

Please find below and/or attached an Office communication concerning this application or proceeding.

Commissioner of Patents and Trademarks

Office Action Summary	Application No. 08/964,388	Applicant(s) Shlomo Touboul
	Examiner Brian H. Shaw	Group Art Unit 2765

☒ Responsive to communication(s) filed on *Nov 6, 1997*

☐ This action is FINAL.

☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11; 453 O.G. 213.

A shortened statutory period for response to this action is set to expire *3* month(s), or thirty days, whichever is longer, from the mailing date of this communication. Failure to respond within the period for response will cause the application to become abandoned. (35 U.S.C. § 133). Extensions of time may be obtained under the provisions of 37 CFR 1.136(a).

Disposition of Claims

☒ Claim(s) *1-70* is/are pending in the application.

Of the above, claim(s) _____ is/are withdrawn from consideration.

☐ Claim(s) _____ is/are allowed.

☒ Claim(s) *1-70* is/are rejected.

☐ Claim(s) _____ is/are objected to.

☐ Claims _____ are subject to restriction or election requirement.

Application Papers

☒ See the attached Notice of Draftsperson's Patent Drawing Review, PTO-948.

☐ The drawing(s) filed on _____ is/are objected to by the Examiner.

☐ The proposed drawing correction, filed on _____ is ☐approved ☐disapproved.

☐ The specification is objected to by the Examiner.

☐ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. § 119

☐ Acknowledgement is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d).

☐ All ☐ Some* ☐ None of the CERTIFIED copies of the priority documents have been

☐ received.

☐ received in Application No. (Series Code/Serial Number) _____.

☐ received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

*Certified copies not received: _____

☐ Acknowledgement is made of a claim for domestic priority under 35 U.S.C. § 119(e).

Attachment(s)

☒ Notice of References Cited, PTO-892

☒ Information Disclosure Statement(s), PTO-1449, Paper No(s). *5*

☐ Interview Summary, PTO-413

☒ Notice of Draftsperson's Patent Drawing Review, PTO-948

☐ Notice of Informal Patent Application, PTO-152

— SEE OFFICE ACTION ON THE FOLLOWING PAGES —

U. S. Patent and Trademark Office
PTO-326 (Rev. 9-95) Office Action Summary Part of Paper No. 6

DETAILED ACTION

Priority

1. Applicant's claim for domestic priority under 35 U.S.C. 119(e) is acknowledged. However, the provisional application upon which priority is claimed fails to provide adequate support under 35 U.S.C. 112 for claims 1-70 of this application.

Claim Objections

2. Claims 23-25 are objected to under 37 CFR 1.75(c) as being in improper form because claim 23 does not refer to a preceding claim. See MPEP § 608.01(n).

Claim Rejections - 35 USC § 112

3. Claim 61 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claim 61 recites the limitation "the logical engine" in line 1. There is insufficient antecedent basis for this limitation in the claim.

Claim Rejections - 35 USC § 103

4. Claims 1, 3, 7-10, 31-35, 55, 56, 63 and 64, are rejected under 35 U.S.C. 103(a) as being unpatentable over McManis in view of Chang et al.

McManis discloses a method and system that determines if received software code was produced by a trusted source. McManis uses digital signatures to verify the source of the software.

McManis does not disclose that the software code should be discarded if it is not verifiable.

Chang et al discloses that if a piece of software cannot be trusted, that the code should be rejected.

It would have been obvious at the time the invention was made to modify the invention of McManis to reject software that originated from an untrusted source.

A person of ordinary skill in the art would have been motivated to make this modification because it could be hazardous to retain software from an untrusted source.

5. Claims 2, 4-6, 11-14, 16 - 22, 29, 30, 36- 45, 46, 52, 53, 54, 57- 59, 60, 61 and 65-70 are rejected under 35 U.S.C. 103(a) as being unpatentable over McManis in view of Chang as applied to claim 1 above, and further in view of Deo.

McManis in view of Chang discloses a system for determining if a piece of software comes from a trusted source.

McManis in view of Chang does not disclose the details of the security rules used.

- As per claims 4-6, and 57-59,

Deo discloses a rules-based security system for a computer. Deo specifies that an example of a "user" to the system could include an applications program. Identification of the "user" is an important component of this and any security system.

It would have been obvious at the time the invention was made to use any "user" identification means available to the security system, for example URL identification, in combination with the verification system of McManis in view of Chang.

A person of ordinary skill in the art would have been motivated to use URL identification in a rules based system because URL's are considered to be reliable and accurate identifiers for information from the Internet.

- As per claims 2, 21, 29, 30, 42, 46, 53, 54, 60, 61, these are all inherent features of the invention of Deo.

- As per claims 16, 39, and 65, assigning an ID to a user or object is an inherent part of any rules based system.

- As per claims 17, 19, 40 and 67-70 Deo provides for controlling a "user's" (i.e., the computer code) access to any controllable system resources, which includes, but is not limited to, reading, writing and communicating, therefore if the "user" is designed to fetch additional components, and the rules state that the "user" is permitted to fetch additional components, then it will.

- As per claims 18, 41 and 66, McManis performs a hashing function on the code.

- As per claims 11-14, 20, 22, 36-38, 43- 45 and 52, Deo system allows for different rules to be defined for different stations and different rules to be defined for combinations of different users, groups and stations and also allows for default rules.

6. Claims 15 and 62 are rejected under 35 U.S.C. 103(a) as being unpatentable over McManis in view of Chang as applied to claim 1 above, and further in view of Official Notice.

McManis in view of Chang discloses a system for determining if a piece of software comes from a trusted source.

McManis in view of Chang does not specifically disclose recording violations in an event log.

Official notice is taken that event logs are commonly used in security programs and they are also common in anti-virus software.

It would have been obvious at the time the invention was made to include an event log when a violation occurs in combination with the invention of McManis in view of Chang.

A person of ordinary skill in the art would have been motivated to make this modification in order for the administrator of the system to monitor specific events and also look for trends.

7. Claims 23-28, 47-49, 50 and 51 are rejected under 35 U.S.C. 103(a) as being unpatentable over McManis in view of Chang as applied to claim 1 above, and further in view of Ji et al.

McManis in view of Chang discloses a system for determining if a piece of software comes from a trusted source.

McManis however does not specifically discuss hostile Downloadable detection.

Ji discloses a method for detecting if an incoming Downloadable is hostile or not. As with most virus checkers, various levels of protection can be defined by the administrator.

It would have been obvious at the time the invention was made to include hostile Downloadable detection (virus checking) capabilities with the verification system of McManis in view of Chang.

A person of ordinary skill in the art would have been motivated to combine the known security ideas of Ji with the known ideas of McManis in view of Chang because combining two security methods would greatly increase system protection.

Conclusion

8. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

- Omura, page 27 "Software Verification and Virus Detection" offers early ideas of software source verification.

- Leach, this PC week article announces that Microsoft plans to use applet signing to verify the source of an applet, and allow the user to establish rules for accepting applets from both trusted and untrusted sources.

- Microsoft Authenticode Technology, this white paper describes how Microsoft's Authenticode product provides protection from potentially hostile downloadables. It provides verification of the source of a piece of code, and allows the user to decide if certain sources are to be trusted, and allows user overside. Digital certificates are used for the verification process.

- Authenticode FAQ, contains a list of answers to frequently asked questions about Microsoft's Authenticode product.

9. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Brian Shaw whose telephone number is (703) 305-0631. The examiner can normally be reached on Monday-Thursday from 7:30 AM to 6:00 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Robert Beausoliel, can be reached on (703) 305-9713 . The fax phone number for the organization where this application or proceeding is assigned is (703) 305-9724.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-3900.

Joseph E. Palys
Primary Examiner

BHS

December 23, 1998

Notice of References Cited

Application No.	Applicant(s)		
08/964,388	Shlomo Touboul		
Examiner		Group Art Unit	Page 1 of 1
Brian H. Shaw		2785	

U.S. PATENT DOCUMENTS

		DOCUMENT NO.	DATE	NAME	CLASS	SUBCLASS
	A	5,692,047	11/25/97	McManis	380	4
	B	5,724,425	03/03/98	Chang et al	380	25
	C	5,720,033	02/07/98	Deo	395	186
	D	5,623,600	04/22/97	Ji et al	395	187.01
	E					
	F					
	G					
	H					
	I					
	J					
	K					
	L					
	M					

FOREIGN PATENT DOCUMENTS

		DOCUMENT NO.	DATE	COUNTRY	NAME	CLASS	SUBCLASS
	N						
	O						
	P						
	Q						
	R						
	S						
	T						

NON-PATENT DOCUMENTS

		DOCUMENT (Including Author, Title, Source, and Pertinent Pages)	DATE
	U	Jim K. Omura, "Novel Applications of Cryptography in Digital Communications", IEEE Communications Magazine, p 27	5/90
	V	Norvin Leach et al, "IE 3.0 applets will earn certification", PC Week, v13, n28, p1(2)	7/96
	W	Microsoft Authenticode Technology, "Ensuring Accountability and Authenticity for Software Components on the Internet", Microsoft Corporation	10/96
	X	Frequently Asked Questions About Authenticode, Microsoft Corporation	2/97

UNITED STATES DEPARTMENT OF COMMERCE
Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.
08/964,488	11/06/97	TOUBUL	40192.00002

LM01/0617

GRAHAM & JAMES
600 HANSEN WAY
PALO ALTO CA 94304-1043

EXAMINER	
SHAW, B	
ART UNIT	PAPER NUMBER
2785	10

DATE MAILED: 06/17/99

Please find below and/or attached an Office communication concerning this application or proceeding.

Commissioner of Patents and Trademarks

PTO-90C (Rev. 2/95) 1- File Copy

Office Action Summary

Application No.	Applicant(s)
08/964,388	Shlomo Touboul
Examiner	**Group Art Unit**
Brian H. Shaw	2765

☒ Responsive to communication(s) filed on *May 3, 1999*.

☒ This action is FINAL.

☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11; 453 O.G. 213.

A shortened statutory period for response to this action is set to expire 3 month(s), or thirty days, whichever is longer, from the mailing date of this communication. Failure to respond within the period for response will cause the application to become abandoned. (35 U.S.C. § 133). Extensions of time may be obtained under the provisions of 37 CFR 1.136(a).

Disposition of Claims

☒ Claim(s) *1-76* is/are pending in the application.

Of the above, claim(s) *23-25, 50, and 67* is/are withdrawn from consideration.

☐ Claim(s) _____ is/are allowed.

☒ Claim(s) *1-22, 26-49, 51-66, and 68-76* is/are rejected.

☐ Claim(s) _____ is/are objected to.

☐ Claims _____ are subject to restriction or election requirement.

Application Papers

☐ See the attached Notice of Draftsperson's Patent Drawing Review, PTO-948.

☐ The drawing(s) filed on _____ is/are objected to by the Examiner.

☐ The proposed drawing correction, filed on _____ is ☐approved ☐disapproved.

☐ The specification is objected to by the Examiner.

☐ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. § 119

☐ Acknowledgement is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d).

☐ All ☐ Some* ☐ None of the CERTIFIED copies of the priority documents have been

☐ received.

☐ received in Application No. (Series Code/Serial Number) _____.

☐ received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

*Certified copies not received: _____.

☒ Acknowledgement is made of a claim for domestic priority under 35 U.S.C. § 119(e).

Attachment(s)

☒ Notice of References Cited, PTO-892

☒ Information Disclosure Statement(s), PTO-1449, Paper No(s). 7

☐ Interview Summary, PTO-413

☐ Notice of Draftsperson's Patent Drawing Review, PTO-948

☐ Notice of Informal Patent Application, PTO-152

— SEE OFFICE ACTION ON THE FOLLOWING PAGES —

U. S. Patent and Trademark Office
PTO-326 (Rev. 9-95) Office Action Summary Part of Paper No. 10

DETAILED ACTION

Specification

1. The abstract of the disclosure is objected to because it is written as two paragraphs. Abstracts should be written as one paragraph. Correction is required. See MPEP § 608.01(b).

Claim Rejections

2. Claim 70 is rejected for having a dependancy to a canceled claim (Claim 67).

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 1, 3, 7-10, 31-35, 55, 56, 63, 64 and 74 are rejected under 35 U.S.C. 103(a) as being unpatentable over McManis in view of Boebert et al.

McManis discloses a method and system that determines if received software code was produced by a trusted source. McManis uses digital signatures to verify the source of the software.

McManis does not disclose that the security elements of the system could be implemented on a separate computer, such as a gateway to the client.

Boebert et al discloses a system where a "secure computer" (or gateway) stands between a client and a public network. The purpose of this secure computer is to monitor all data exchanges and command requests between the public network and the client.

It would have been obvious at the time the invention was made to modify the invention of McManis to locate the security components of McManis to an a gateway server, as Boebert et al does.

A person of ordinary skill in the art would have been motivated to make this modification because, as Boebert et al shows, centralization of security mechanisms adds the benefit of simplifying administration of security databases (Boebert et al, Column 30, lines 17-20). Additional benefits of centralization are listed in Boebert et al, Column 29, Line 56 - Column 30, Line 38.

5. Claims 2, 4-6, 11-14, 16-22, 29, 30, 36-46, 52-54, 57-61, 65, 66, 68, 69, 75 and 76 are rejected under 35 U.S.C. 103(a) as being unpatentable over McManis in view of Boebert et al as applied to claim 1 above, and further in view of Deo.

McManis in view of Boebert et al discloses a system for determining if a piece of software comes from a trusted source.

McManis in view of Boebert et al does not disclose the details of the security rules used.

- As per claims 4-6, and 57-59,

Deo discloses a rules-based security system for a computer. Deo specifies that an example of a "user" to the system could include an applications program. Identification of the "user" is an important component of this and any security system.

It would have been obvious at the time the invention was made to use any "user" identification means available to the security system, for example URL identification, in combination with the verification system of McManis in view of Chang.

A person of ordinary skill in the art would have been motivated to use URL identification in a rules based system because URL's are considered to be reliable and accurate identifiers for information from the Internet.

- As per claims 2, 21, 29, 30, 42, 46, 53, 54, 60 and 61, these are all inherent features of the invention of Deo.

- As per claims 16 and 39, assigning an ID to a user or object is an inherent part of any rules based system.

- As per claims 17, 19, 40, 65, 68 and 69, Deo provides for controlling a "user's" (i.e., the computer code) access to any controllable system resources, which includes, but is not limited to, reading, writing and communicating, therefore if the "user" is designed to fetch additional

components, and the rules state that the "user" is permitted to fetch additional components, then it will.

- As per claims 18, 41, 66, 75 and 76, McManis performs a hashing function on the code.

- As per claims 11-14, 20, 22, 36-38, 43- 45 and 52, Deo system allows for different rules to be defined for different stations and different rules to be defined for combinations of different users, groups and stations and also allows for default rules.

6. Claims 15 and 62 are rejected under 35 U.S.C. 103(a) as being unpatentable over McManis in view of Boebert et al as applied to claim 1 above, and further in view of Official Notice.

McManis in view of Boebert et al discloses a system for determining if a piece of software comes from a trusted source.

McManis in view of Boebert et al does not specifically disclose recording violations in an event log.

Official notice is taken that event logs are commonly used in security programs and they are also common in anti-virus software.

It would have been obvious at the time the invention was made to include an event log when a violation occurs in combination with the invention of McManis in view of Boebert et al.

A person of ordinary skill in the art would have been motivated to make this modification in order for the administrator of the system to monitor specific events and also look for trends.

7. Claims 26-28, 50, 51 and 71-73 are rejected under 35 U.S.C. 103(a) as being unpatentable over McManis in view of Boebert et al as applied to claim 1 above, and further in view of Ji et al.

McManis in view of Boebert et al discloses a system for determining if a piece of software comes from a trusted source.

McManis however does not specifically discuss hostile Downloadable detection.

Ji discloses a method for detecting if an incoming Downloadable is hostile or not. As with most virus checkers, various levels of protection can be defined by the administrator.

It would have been obvious at the time the invention was made to include hostile Downloadable detection (virus checking) capabilities with the verification system of McManis in view of Boebert et al.

A person of ordinary skill in the art would have been motivated to combine the known security ideas of Ji with the known ideas of McManis in view of Boebert et al because combining two security methods would greatly increase system protection.

8. Claims 47-49 are rejected under 35 U.S.C. 103(a) as being unpatentable over McManis in view of Boebert et al in view of Deo as applied to claim 46 above, and further in view of Ji et al.

McManis in view of Boebert et al in view of Deo discloses a system for determining if a piece of software comes from a trusted source.

McManis however does not specifically discuss hostile Downloadable detection.

Ji discloses a method for detecting if an incoming Downloadable is hostile or not. As with most virus checkers, various levels of protection can be defined by the administrator.

It would have been obvious at the time the invention was made to include hostile Downloadable detection (virus checking) capabilities with the verification system of McManis in view of Boebert et al.

A person of ordinary skill in the art would have been motivated to combine the known security ideas of Ji with the known ideas of McManis in view of Boebert et al because combining two security methods would greatly increase system protection.

Conclusion

9. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL.** See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period,

will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

10. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Brian Shaw whose telephone number is (703) 305-0631. The examiner can normally be reached on Tuesday - Friday from 7:30 AM to 6:00 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Robert Beausoliel, can be reached on (703) 305-9713 . The fax phone number for the organization where this application or proceeding is assigned is (703) 305-3718 .

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-3900.

BHS

June 11, 1999

ROBERT W. BEAUSOLIEL, JR.
SUPERVISORY PATENT EXAMINER
GROUP 2700

EXPRESS MAIL LABEL NO: EL...875714US

Our Docket No. 40492.00002

#14/B

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re Application of: Shlomo Touboul			
Serial No.: 08/964,388		Examiner:	B. Shaw
Filed: November 6, 1997			
For:	System and Method for Protecting a Computer and a Network from Hostile Downloadables	Art Unit:	2785

BOX - CPA
Assistant Commissioner for Patents
Washington, D.C. 20231

PRELIMINARY AMENDMENT

Sir:

This preliminary amendment addresses the rejections of the final office action mailed on June 17, 1999, the statutory period for response in the parent application being extended until November 17, 1999 by the enclosed 2-month petition for extension of time. Before considering the subject application, please amend it as follows:

IN THE SPECIFICATION:

On page 11 line 14, after "Downloadable," delete "the".

IN THE ABSTRACT:

Please remove the paragraph break.

131/195238.01.00
102699/0932/40482.00002

EXPRESS MAIL LABEL NO: EL005875714US

Our Docket No. 40492.00002

IN THE CLAIMS:

1. (Twice amended) A computer-based method, comprising the steps of:

receiving an incoming Downloadable addressed to a client, by a server that serves as a gateway to the client;

comparing, by the server, [at least a portion of] Downloadable security profile data pertaining to the Downloadable against a security policy to determine if the security policy has been violated; and

preventing execution of the Downloadable by the client if the security policy has been violated.

2. (Once amended) The method of claim 1, further comprising the step[s] of decomposing the Downloadable into the Downloadable security profile data[, and comparing the Downloadable security profile data against the security policy].

~~32.~~ 31. (Twice amended) A system for execution by a server that serves as a gateway to a client, the system comprising:

a security policy;

an interface for receiving an incoming Downloadable addressed to a client;

a comparator, coupled to the interface, for [applying] comparing Downloadable security profile data pertaining to the Downloadable against the security policy [to the Downloadable] to determine if the security policy has been violated; and

a logical engine for preventing execution of the Downloadable by the client if the security policy has been violated.

~~54.~~ 53. (Once amended) The system of claim ~~32~~ 31, further comprising a code scanner coupled to the comparator for decomposing the Downloadable into the Downloadable security profile data.

64 63. (Twice amended) A system for execution on a server that serves as a gateway to a client, comprising:

means for receiving an incoming Downloadable addressed to a client;

means for comparing Downloadable security profile data pertaining to the Downloadable against a security policy to determine if the security policy has been violated; and

means for preventing execution of the Downloadable by the client if the security policy has been violated.

65 64. (Twice amended) A computer-readable storage medium storing program code for causing a server that serves as a gateway to a client to perform the steps of:

receiving an incoming Downloadable addressed to a client;

comparing Downloadable security profile data pertaining to the Downloadable against a security policy to determine if the security policy has been violated; and

preventing execution of the Downloadable by the client if the security policy has been violated.

65. (Twice amended) A computer-based method for generating a Downloadable ID to identify a Downloadable, comprising the steps of:

obtaining a Downloadable;

fetching, if the Downloadable includes one or more references to a component, at least one component identified by the one or more references; and

performing a function on the Downloadable and all components fetched to generate a Downloadable ID.

70. (Twice amended) The method of claim [67] 65, wherein the step of fetching includes fetching all components referenced by the Downloadable.

EXPRESS MAIL LABEL NO EL638875714US

Our Docket No. 40492.00002

Please add the following claims:

77. The method of claim 1, further comprising the steps of recognizing the incoming Downloadable, and obtaining the Downloadable security profile data for the incoming Downloadable from memory.

78. The system of claim 31, further comprising memory storing the Downloadable security profile data for the incoming Downloadable.--

REMARKS

Claims 1-22, 26-59, 61-66 and 68-76 were examined and rejected in this case. Claims 1, 2, 31, 53, 63-65 and 70 are being amended. Claims 77 and 78 are being added. Claims 1-22, 26-59, 61-66 and 68-78 are currently pending. Reconsideration of the application as amended is respectfully requested.

Applicant requests the Examiner to enter the above amendment to the Specification. No new matter is being added.

In paragraph 1, the Examiner objected to the Abstract because it was written as two paragraphs. Accordingly, Applicant is amending the Abstract to remove the paragraph break.

In paragraph 2, the Examiner rejected claim 70 for being dependent on a canceled claim. Accordingly, Applicant is amending claim 70 to depend from pending claim 65.

In paragraph 4, the Examiner rejected claims 1, 3, 7-10, 31-35, 55, 56, 63, 64 and 74 under 35 USC § 103(a) as being unpatentable McManis in view of Boebert. McManis teaches examining digital signatures by the originating party, by the compiling party, and by the executing party. Boebert teaches secure transfer of data through a secure computer to clients. However, claims 1, 31, 63, 64 and 74, as amended, similarly recite "comparing, by the server, Downloadable security profile data pertaining to the Downloadable against a security policy to determine if the security policy has been violated." On page 12 lines 17-20, the Specification describes an embodiment wherein

the particular Downloadable security profile data 310 includes "the list of all potentially hostile or suspicious computer operations that may be attempted by a specific Downloadable 307, and may also include the respective arguments of these operations." Applicant respectfully submits that neither McManis nor Boebert teaches this step of comparing potentially hostile operations in a Downloadable against a security policy as recited in independent claims 1, 31, 63, 64 and 74. For at least the same reasons, Applicant respectfully submits that claims 3, 7-10, 32-35, 55, 56, dependent therefrom, are also patentable.

In paragraph 5, the Examiner rejected claims 2, 4-6, 11-14, 16-22, 29, 30, 36-46, 52-54, 57-61, 65, 66, 68, 69, 75 and 76 under 35 USC § 103(a) as unpatentable over McManis in view of Boebert and further in view of Deo. In subparagraph 5 of paragraph 5, the Examiner cited that any "user" identification would be obvious over McManis in view of Chang. Since the overall rejection is based on McManis, Boebert and Deo, Applicant is unsure whether the Examiner intended to cite Boebert instead of Chang. For the purposes of this response, Applicant will assume that the Examiner intended to cite Boebert. Applicant requests clarification.

Deo teaches a security platform using object-oriented rules for UNIX operating systems. As stated above, independent claims 1, 31, 63, 64 and 74, as amended, similarly recite "comparing, by the server, Downloadable security profile data pertaining to the Downloadable against a security policy to determine if the security policy has been violated." Independent claim 76 also includes this limitation. Like McManis and Boebert, Applicant respectfully submits that Deo does not teach this step. Accordingly, for at least these reasons, Applicant respectfully submits that claims 2, 4-6, 11-14, 16-22, 29, 30, 36-46, 52-54, 57-61 and 75, dependent therefrom, and independent claim 76 are patentable over McManis in view of Boebert and further in view of Deo.

Independent claim 65 recites a method for generating a Downloadable ID to identify the incoming Downloadable. Neither McManis, Boebert nor Deo teach this method. Accordingly, Applicant respectfully submits that claim 65, and for at least this

reason claims 66, 68, 69, dependent therefrom, are patentable over McManis in view of Boebert and further in view of Deo.

In paragraph 6, the Examiner rejected claims 15 and 62 under 35 USC § 103(a) as unpatentable over McManis in view of Boebert and further in view of Official Notice. The Examiner asserted that event logs are known in the art. However, since claims 15 and 62 depend from claims 1 and 31, respectively, Applicant respectfully submits that claims 15 and 62 are patentable for at least the same reasons.

In paragraph 7, the Examiner rejected claims 26-28, 50, 51 and 71-73 under 35 § 103(a) as unpatentable over McManis in view of Boebert and further in view of Ji. The Examiner admitted that McManis and Boebert do not teach hostile Downloadable detection, and asserted that Ji teaches it. Applicant respectfully traverses. Ji teaches gateway detection of viruses attached to executable files, and does not teach hostile Downloadable detection. As is well known in the art, a Downloadable is mobile code that is requested by an ongoing process, downloaded from a source computer to a destination computer for automatic execution. The programs or documents of Ji are not Downloadables. Further, Ji does not teach a "comparing, by the server, Downloadable security profile data pertaining to the Downloadable against a security policy to determine if the security policy has been violated," as recited in independent claims 1 and 31. Since claims 26-28 and 71-73 depend from claim 1 and claims 50 and 51 depend from claim 31, Applicant respectfully submits that claims 26-28, 50, 51 and 71-73 are patentable for at least the same reasons.

In paragraph 8, the Examiner rejected claims 47-49 under 35 USC § 103(a) as unpatentable over McManis, in view of Boebert, further in view of Deo and still further in view of Ji. For at least the reasons discussed above, Applicant respectfully submits that claims 47-49 are patentable.

Applicant respectfully requests that the Examiner withdraw the rejection of claims 1-22, 26-59, 61-66 and 68-76.

EXPRESS MAIL LABEL NO: EL058875714US

Our Docket No. 40492.00002

If the Examiner has any questions or needs any additional information, the Examiner is invited to telephone the undersigned attorney at (650) 843-3392.

If for any reason an insufficient fee has been paid, the Assistant Commissioner is hereby authorized to charge the insufficiency to Deposit Account No. 05-0150.

Respectfully Submitted,
Shlomo Touboul

Dated: 10-27-99

Graham & James LLP
600 Hansen Way
Palo Alto, CA 94304-1043
650-856-6500

Marc A. Sockol
Attorney for Applicants
Reg. No. 40,823

INFORMATION DISCLOSURE CITATION
IN AN APPLICATION
PTO 1449
(Use several sheets if necessary)

DOCKET NUMBER	APPLICATION NUMBER
40492.00002	08/964,388
APPLICANT: Shlomo Touboul	
FILING DATE: November 6, 1997	GROUP ART UNIT: 2785

U.S. PATENT DOCUMENTS

EXAMINER INITIAL	REF	DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE APPROPRIAT
	A	5,983,348	09/10/97	Shuang Ji	713	200	
	B	5,859,966	10/10/95	Kenneth J. Hayman et al.	395	186	
	C	5,866,481	02/06/97	James E. Walsh et al.	395	186	
	D	5,359,659	08/19/92	Doren Rosenthal	380	4	
	E						
	F						
	G						

FOREIGN PATENT DOCUMENTS

	REF	DOCUMENT NUMBER	DATE	COUNTRY	CLASS	SUBCLASS	TRANSLATIO YES	TRANSLATIO NO
	H							
	I							

OTHER DOCUMENTS *(Including Author, Title, Date, Pertinent Pages, Etc.)*

	J	Zhang, X.N., Computer, "Secure Code Distribution," Vol. 30, June, 1997, Pages: 76 - 79
	K	
	L	
	M	
	N	
	O	
	P	
	Q	
	S	

EXAMINER: Clifton Mul — DATE CONSIDERED: 6/7/05

EXAMINER: Initial if citation considered, whether or not citation is in conformance with MPEP Section 609; Draw line through citation if not information and not considered. Include copy of this with next communication to applicant.

Sheet 1 of

131/200765.01
032000/1103/40492.00002

#6A/8-7-03 V. Jones

Attorney's Docket No.: 43426.00011 *PATENT*

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF MAILING

I hereby certify that this paper (along with any paper referred to as being attached or enclosed) is being deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on

Date: July 31, 2003 By: Eileen M. Janikowski

In Re Patent Application of:

Shlomo Touboul

Application No: 09/539,667

Filed: March 30, 2000

For: SYSTEM AND METHOD FOR PROTECTING A COMPUTER AND A NETWORK FROM HOSTILE DOWNLOADABLES

Examiner: Christopher A. Revak

Art Unit: 2131

Assistant Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

AMENDMENT AND RESPONSE TO OFFICE ACTION UNDER 37 C.F.R. §1.111

Sir:

In response to the Office Action dated July 1, 2003 and pursuant to 37 C.F.R. §1.111, applicant respectfully requests that the above-identified application be amended as follows:

IN THE ABSTRACT OF THE DISCLOSURE:

Kindly replace the Abstract of the Disclosure with the following text:

-- A computer-based method for generating a Downloadable ID to identify a Downloadable, including obtaining a Downloadable that includes one or more references to software components required by the Downloadable, fetching at least one software component identified by the one or more references, and performing a function on the Downloadable and the fetched software components to generate a Downloadable ID. A system and a computer-readable storage medium are also described and claimed. --

IN THE CLAIMS:

Kindly cancel claims 9 and 19 without prejudice.

Please substitute the following claims for the pending claims with the same number:

1. (Currently amended) A computer-based method for generating a Downloadable ID to identify a Downloadable, comprising [the steps of]:

obtaining a Downloadable that includes one or more references to software components required by the Downloadable;

fetching[, if the Downloadable includes one or more references to a component,] at least one software component identified by the one or more references; and

performing a function on the Downloadable and [all] the fetched software components [fetched] to generate a Downloadable ID.

2. (Original) The method of claim 1, wherein the Downloadable includes an applet.

3. (Currently amended) The method of claim 1, wherein the Downloadable includes an [ActiveX™] active software control.

4. (Original) The method of claim 1, wherein the Downloadable includes a plugin.

5. (Original) The method of claim 1, wherein the Downloadable includes HTML code.

6. (Original) The method of claim 1, wherein the Downloadable includes an application program.

7. (Original) The method of claim 1, wherein the function includes a hashing function.

8. (Currently amended) The method of claim 1, wherein [the step of] said fetching includes [the step of] fetching the first software component referenced by the Downloadable.

9. (Cancelled)

10. (Currently amended) The method of claim 1, wherein [the step of] said fetching includes fetching all software components referenced by the Downloadable.

11. (Currently amended) A system for generating a Downloadable ID to identify a Downloadable, comprising:

a communications engine for obtaining a Downloadable that includes one or more references to software components required by the Downloadable; and

an ID generator coupled to the communications engine for fetching[, if the Downloadable includes one or more references to a component,] at least one software component identified by the one or more references, and for performing a function on the Downloadable and [all] the fetched software components [fetched] to generate a Downloadable ID.

12. (Original) The system of claim 11, wherein the Downloadable includes an applet.

13. (Currently amended) The system of claim 11, wherein the Downloadable includes an [ActiveX™] active software control.

14. (Original) The system of claim 11, wherein the Downloadable includes a plugin.

15. (Original) The system of claim 11, wherein the Downloadable includes HTML code.

16. (Original) The system of claim 11, wherein the Downloadable includes an application program.

17. (Original) The system of claim 11, wherein the function includes a hashing function.

18. (Currently amended) The system of claim 11, wherein the ID generator fetches the first software component referenced by the Downloadable.

19. (Cancelled)

20. (Currently amended) The method of claim 11, wherein the ID generator fetches all software components referenced by the Downloadable.

21. (Currently amended) A system for generating a Downloadable ID to identify a Downloadable, comprising:

means for obtaining a Downloadable that includes one or more references to software components required by the Downloadable;

means for fetching[, if the Downloadable includes one or more references to a component,] at least one software component identified by the one or more references; and

means for performing a function on the Downloadable and [all] the fetched software components [fetched] to generate a Downloadable ID.

22. (Currently amended) A computer-readable storage medium storing program code for causing a computer to perform the steps of:

obtaining a Downloadable that includes one or more references to software components required by the Downloadable;

fetching[, if the Downloadable includes one or more references to a component,] at least one software component identified by the one or more references; and

performing a function on the Downloadable and [all] the fetched software components [fetched] to generate a Downloadable ID.

REMARKS

Applicant has carefully studied the outstanding Office Action. The present amendment is intended to place the application in condition for allowance and is believed to overcome all of the objections and rejections made by the Examiner. Favorable reconsideration and allowance of the application are respectfully requested.

Applicant has canceled claims 9 and 19, and amended claims 1, 3, 8, 10, 11, 13, 18 and 20 - 22 to more properly claim the present invention. No new matter has been added. Claims 1 – 8, 10 – 18 and 20 – 22 are presented for examination.

Applicant notes that the page headers of the Office Action indicate an incorrect Application/Control Number.

In paragraphs 2 and 3 of the Office Action, the Examiner has objected to the abstract of the disclosure. Accordingly, applicant has amended the abstract so as to conform to the proper language and format.

In paragraphs 4 and 5 of the Office Action, the Examiner has rejected claims 1, 11, 21 and 22 under the judicially created doctrine of double patenting. Accordingly, applicant is submitting a terminal disclaimer with the present amendment.

In paragraphs 6 and 7 of the Office Action, the Examiner has rejected claims 3 and 13 under 35 U.S.C. §112, second paragraph as being indefinite. Applicant has amended these claims accordingly.

In paragraphs 8 and 9 of the Office Action, the Examiner has rejected claims 1, 7, 8, 10, 11, 17, 18, and 20 – 22 under 35 U.S.C. §102(e) as being anticipated by Apperson et al., U.S. Patent No. 5,978,484 ("Apperson").

In paragraphs 10 and 11 of the Office Action, the Examiner has rejected claims 2 – 4 and 12 – 14 under 35 U.S.C. §103(a) as being unpatentable over Apperson in view of Khare, "*Microsoft Authenticode Analyzed*", July 22, 1996, xent.com/FoRK-archive/summer96/0338.html, pg. 1 and 2 ("Khare").

In paragraph 12 of the Office Action, the Examiner has rejected claims 5, 6, 9, 15, 16 and 19 under 35 U.S.C. §103(a) as being unpatentable over Apperson. Applicant has canceled claims 9 and 19 without acquiescence to the Examiner's reasons for rejection and respectfully submits that rejection of those claims is thus rendered moot.

Distinctions between Claimed Invention and U.S. Patent No. 5,978,484 to Apperson et al in view of Khare, "Microsoft Authenticode Analyzed", July 22, 1996, xent.com/FoRK-archive/summer96/0338.html, pg. 1 and 2

The present invention concerns generation of an ID for mobile code downloaded to a client computer, referred to as a Downloadable. Specifically, the present invention fetches software components required by the Downloadable, and performs a hashing function on the Downloadable together with its fetched components (original specification / page 3, lines 11 – 14; page 15, lines 21 – 24; page 19, line 21 – page 20, line 6; FIG. 8). Thus, for a Java applet, the present invention fetches Java classes identified by the applet bytecode, and generates the Downloadable ID from the applet and the fetched Java classes; and for an ActiveX™ control, the present invention fetches components listed in its .INF file, and generates a Downloadable ID from the ActiveX™ control and the fetched components (original specification / page 9, lines 15 – 18).

An advantage of the present invention is that it produces the same ID for a Downloadable, regardless of which software components are included with the Downloadable and which software components are only referenced (original specification / page 9, lines 18 – 20; page 20, lines 5 and 6). The same Downloadable may be delivered with some required software components included and others missing, and in each case the generated Downloadable ID will be the same. Thus the same Downloadable is recognized through many equivalent guises.

Apperson describes use of digital certificates to authorize privileges for executable code, such as file I/O privileges, network privileges and registry privileges (Apperson / col. 2, lines 41 – 53; col. 4, lines 33 – 43; FIG. 2).

Khare describes Microsoft Corporation's implementation of digital signatures, referred to as Authenticode, as applied to ActiveX controls and Java applets.

In distinction to the present invention, Apperson and Khare do not teach fetching software components of executable code. In order to further clarify this distinction, applicant has amended the claims so as to refer to software components required by the Downloadable.

In paragraph 9 of the Office Action, the Examiner has indicated that Apperson discloses fetching components of a Downloadable. Applicant respectfully submits that Apperson's privilege request code does not include components of a Downloadable, but instead includes a list of "*privileges or*

privilege categories that the executable code might perform on the client machine" (Apperson / col. 2, lines 45 – 47).

The rejections of claims 1 –8 and 10 in paragraphs 8 - 12 of the Office Action will now be dealt with specifically.

As to amended independent method claim 1, applicant respectfully submits that the limitation in claim 1 of:

"*fetching at least one software component identified by the one or more references*"

is neither shown nor suggested in Apperson or Khare.

Because claims 2 – 8 and 10 depend from claim 1 and include additional features, applicant respectfully submits that claims 2 - 8 and 10 are not anticipated or rendered obvious by Apperson and Khare, taken alone or in combination.

Accordingly claims 1 – 8 and 10 are deemed to be allowable.

As to amended independent system claim 11, applicant respectfully submits that the limitation in claim 11 of:

"*an ID generator coupled to the communications engine for fetching at least one software component identified by the one or more references*"

is neither shown nor suggested in Apperson or Khare.

Because claims 12 – 18 and 20 depend from claim 11 and include additional features, applicant respectfully submits that claims 12 – 18 and 20 are not anticipated or rendered obvious by Apperson and Khare, taken alone or in combination.

Accordingly claims 12 – 18 and 20 are deemed to be allowable.

As to amended independent system claim 21, applicant respectfully submits that the limitation in claim 21 of:

"*means for fetching at least one software component identified by the one or more references*"

is neither shown nor suggested in Apperson or Khare.

Accordingly claim 21 is deemed to be allowable.

As to amended independent system claim 22, applicant respectfully submits that the limitation in claim 22 of:

"*fetching at least one software component identified by the one or more references*"

is neither shown nor suggested in Apperson or Khare.

Accordingly claim 22 is deemed to be allowable.

Support for Amended Claims in Original Specification

Regarding amended claims 1, 8, 10, 11, 18 and 20 - 22, fetching software components is described in the original specification on page 9, lines 13 – 18 and FIG. 8.

For the foregoing reasons, applicant respectfully submits that the applicable objections and rejections have been overcome and that the claims are in condition for allowance.

If the Examiner has any questions or needs any additional information, the Examiner is invited to telephone the undersigned attorney at (650) 843-3392. If for any reason an insufficient fee has been paid, please charge the insufficiency to Deposit Account No. 05-0150.

Date: July 31, 2003

Respectfully submitted,

Squire, Sanders & Dempsey L.L.P.
600 Hansen Way
Palo Alto, CA 94304-1043
Telephone: (650) 856-6500
Facsimile: (650) 843-8777

By: 
Marc A. Sockol
Attorney for Applicant
Registration No. 40,823

Copy of #14

S/N 09/024,576 PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Richard R. Viets et al.	Examiner: Nguyen Xuan Nguyen
Serial No.:	09/024,576	Group Art Unit: 2787
Filed:	February 17, 1998	Docket: 105.140US1
Title:	SYSTEM AND METHOD FOR CONTROLLING ACCESS TO DOCUMENTS STORED ON AN INTERNAL NETWORK	

RESPONSE UNDER 37 CFR § 1.111

Commissioner for Patents
Washington, D.C. 20231

Applicant has carefully reviewed and considered the Office Action mailed on June 1, 2000, and the references cited therewith. Thirty-seven claims remain pending in this application.

Summary of the Invention

The present invention is a system and method for permitting limited access to documents stored on an internal company network. Clients are assigned to one or more roles. For instance, a company may decide to limit access to documents depending on whether a reseller handles domestic or international sales. Documents relevant to an international reseller may not be pertinent to a domestic reseller, and vice versa. In addition, it may be advantageous to wall off one role from another. Domestic resellers and engineers do not need to access sensitive marketing material regarding penetration of foreign markets; marketing and the international reseller do need such access.

According to the present invention, in one embodiment document access is limited to the roles identified as requiring access to that document. Such an approach is more flexible than a user-based access control system and can be used to provide a fine degree of granularity across the document space. If a user is added to the system, one needs only to assign one or more roles to that user to provide access to all relevant documents. When a user attempts to access a controlled document, a check is made to determine if someone assigned the user's roles is allowed to access the document. If so, the document is retrieved and presented to the user.

In one embodiment, as is illustrated in Fig. 2 and discussed at p. 5, line 13 through p. 10, line 9, the document control server receives a document request from the external interface for a first document. The document control server determines a user associated with the document request and authenticates the user. The system then checks to see if the user is authorized to access the document requested. If so, the system retrieves the document from the document server, cleans the document and forwards the clean version of the document to the user. As is noted at p. 8, line 11 through p. 10, line 9, "cleaning" involves replacing embedded links within each document and redirecting page access as necessary to preserve the confidentiality of file structure within the intranet.

The approach described by applicant permits limited and controlled access to the internal network, while preventing the security problems associated with allowing business partners unfettered, or more coarsely controlled, access to the internal network. It also avoids the headaches mentioned in the Background of maintaining a mirror image of relevant documents outside the internal network.

§103 Rejection of the Claims

1) The Applicable Law

The Examiner has the burden under 35 U.S.C. § 103 to establish a *prima facie* case of obviousness. *In re Fine*, 837 F.2d 1071, 1074, 5 USPQ2d 1596, 1598 (Fed. Cir. 1988). To do that the Examiner must show that some objective teaching in the prior art or some knowledge generally available to one of ordinary skill in the art would lead an individual to combine the relevant teachings of the references. *Id.*

The court in *Fine* stated that:

> Obviousness is tested by "what the combined teaching of the references would have suggested to those of ordinary skill in the art." *In re Keller*, 642 F.2d 413, 425, 208 USPQ 871, 878 (CCPA 1981)). But it "cannot be established by combining the teachings of the prior art to produce the claimed invention, absent some teaching or suggestion supporting the combination." *ACS Hosp. Sys.*, 732 F.2d at 1577, 221 USPQ at 933. And "teachings of references can be combined *only* if there is some suggestion or incentive to do so."

Id. (emphasis in original).

The M.P.E.P. adopts this line of reasoning, stating that

> To establish a *prima facie* case of obviousness, three base criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, and not based on applicant's disclosure.

M.P.E.P. § 2142 (citing *In re Vaeck*, 947 F.2d 488, 20 USPQ2d 1438 (Fed.Cir. 1991)).

An invention can be obvious even though the suggestion to combine prior art teachings is not found in a specific reference. *In re Oetiker*, 24 USPQ2d 1443 (Fed. Cir. 1992). At the same time, however, although it is not necessary that the cited references or prior art specifically suggest making the combination, there must be a teaching within the knowledge generally available to one of ordinary skill in the art at the time of the invention which provides the suggestion or motivation to combine prior art teachings and which applies that combination to solve the same or similar problem which the claimed invention addresses. *M.P.E.P.* § 2143.01. One of ordinary skill in the art will be presumed to know of any such teaching. (See, e.g., *In re Nilssen*, 851 F.2d 1401, 1403, 7 USPQ2d 1500, 1502 (Fed. Cir. 1988) and *In re Wood*, 599 F.2d 1032, 1037, 202 USPQ 171, 174 (CCPA 1979)).

The test for obviousness under § 103 must take into consideration the invention as a whole; that is, one must consider the particular problem solved by the combination of elements that define the invention. *Interconnect Planning Corp. v. Feil*, 774 F.2d 1132, 1143, 227 USPQ 543, 551 (Fed. Cir. 1985). Furthermore, claims must be interpreted in light of the specification, claim language, other claims and prosecution history. *Panduit Corp. v. Dennison Mfg. Co.*, 810 F.2d 1561, 1568, 1 USPQ2d 1593, 1597 (Fed. Cir. 1987), *cert. denied*, 481 U.S. 1052 (1987). At the same time, a prior patent cited as a § 103 reference must be considered in its entirety, "*i.e.* as a *whole*, including portions that lead away from the invention." *Id.* That is, the Examiner must,

as one of the inquiries pertinent to any obviousness inquiry under 35 U.S.C. § 103, recognize and consider not only the similarities but also the critical differences between the claimed invention and the prior art. *In re Bond*, 910 F.2d 831, 834, 15 USPQ2d 1566, 1568 (Fed. Cir. 1990), *reh'g denied*, 1990 U.S. App. LEXIS 19971 (Fed. Cir. 1990). Finally, the Examiner must avoid hindsight. *Id.*

As noted in the response to the first Office Action, the Examiner rejected a number of claims as being unpatentable over a single reference, U.S. Patent No. 5,826,029 issued October 20, 1998 to Gore, Jr (hereinafter "Gore"). As noted above, the Examiner can, as part of establishing a *prima facie* case of obviousness, show that some knowledge generally available to one of ordinary skill in the art would lead an individual to combine the relevant teaching of the references. *Fine*, 837 F.2d at 1074, 5 USPQ2d 1596 at 1598. Where the single reference shows some but not all the elements of the claims, the Examiner can supplement that teaching with what was common knowledge in the art at the time the invention was made.

> The rationale supporting an obviousness rejection may be based on common knowledge in the art or "well-known" prior art. The examiner may take official notice of facts outside of the record which are capable of instant and unquestionable demonstration as being "well- known" in the art. In re Ahlert, 424 F.2d 1088, 165 USPQ 418, 420 (CCPA 1970) (Board properly took judicial notice that "it is common practice to postheat a weld after the welding operation is completed" and that "it is old to adjust the intensity of a flame in accordance with the heat requirements."). See also In re Seifreid, 407 F.2d 897, 160 USPQ 804 (CCPA 1969) (Examiner's statement that polyethylene terephthalate films are commonly known to be shrinkable is a statement of common knowledge in the art, supported by the references of record.).
>
> If justified, the examiner should not be obliged to spend time to produce documentary proof. If the knowledge is of such notorious character that judicial notice can be taken, it is sufficient so to state. In re Malcolm, 129 F.2d 529, 54 USPQ 235 (CCPA 1942). If the applicant traverses such an assertion the examiner should cite a reference in support of his or her position.
>
> If applicant does not seasonably traverse the well known statement during examination, then the object of the well known statement is taken to be admitted prior art. In re Chevenard, 139 F.2d 71, 60 USPQ 239 (CCPA 1943). A seasonable challenge constitutes a demand for evidence made as soon as practicable during prosecution. Thus, applicant is charged with rebutting the well known statement in the next response after the Office Action in which the well

known statement was made. This is necessary because the examiner must be given the opportunity to provide evidence in the next Office Action or explain why no evidence is required. If the examiner adds a reference to the rejection in the next action after applicant's rebuttal, the newly cited reference, if it is added merely as evidence of the prior well known statement, does not result in a new issue and thus the action can potentially be made final. If no amendments are made to the claims, the examiner must not rely on any other teachings in the reference if the rejection is made final.

When a rejection is based on facts within the personal knowledge of the examiner, the data should be stated as specifically as possible, and the facts must be supported, when called for by the applicant, by an affidavit from the examiner. Such an affidavit is subject to contradiction or explanation by the affidavits of the applicant and other persons. See 37 CFR 1.107.

M.P.E.P. § 2144.03.

2) ***Discussion of the Rejections***

I) The rejection of claims 1-6 and 35 under 35 U.S.C. § 103 based on Gore, Mighdoll and Shieh

Claims 1-6, 11 and 35 were rejected under 35 U.S.C. 103(a) as being unpatentable over Gore, Jr. et al. (U.S. Patent No. 5,826,029) in view of Mighdoll et al. (U.S. Patent No. 5,918,013) and further in view of Shieh et al. (U.S. Patent No. 5,933,600).

Applicant respectfully traverses the rejection of claims 1-6, 11 and 35. It is respectfully submitted that Gore, Jr., Mighdoll and Shieh, alone or combined, do not teach or suggest all of the elements of the claimed invention. The Examiner's rejection of claim 11 used the disclosures of Gore and Mighdoll only. Claim 11 will, therefore, be discussed in Section III below with the rejections of claims 12, 15, and 16, which also were rejected under 35 U.S.C. 103(a) as being unpatentable over Gore, Jr. et al. in view of Mighdoll et al.

Claims 1-6 and 35 include, among other things, building a client list that includes assigning each client a role. The Examiner states that neither Gore Jr., nor Mighdoll teach building a client list but that Shieh teaches a comparison between the selected requestor list and a stored set of unallowed links. The Examiner states that one of ordinary skill in the art at the time of the invention would take Shieh's comparison of a requestor list and unallowed links and use it

with Gore to build Applicant's system. That is, to build a client list in which each client has a role and to use that role to limit the documents that can be accessed by a particular user.

Shieh's selected requestor list is not, however, a client list as that term is defined by applicant. Shieh talks about limiting access to links at a client level in a client/server architecture (See Shieh, col. 3, lines 44-60). This is done by "altering the client system's 10 or server system's 12 programming to perform a comparison between the selected requestor list and a stored set of unallowed links." *Shieh*, col.3, lines 51-53. Shieh, therefore, acts more as a site filter than as a system for limiting access to documents. In addition, since the modifications are made to the client system or server system as described above, there is no filtering based on the user. That is, a user at one blocked client system can go to another client system that is not blocked to gain the desired access.

Applicant, on the other hand, determines the client's identity, determines the role or roles available to that client and limits access to documents based on those roles. Applicant's use of the term "client" is not to be confused with the use of the term "client system" in a client server architecture such as described by Shieh. Instead, Applicant's clients are the system users (see, e.g., p. 6, line 18 through p. 7, line 15).

As note above, to establish a *prima facie* case of obviousness, the Examiner must show "some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings." *M.P.E.P.* § 2142. The Examiner states that

> it would have been obvious to a person of ordinary skill in the art at the time of the invention to employ Shieh's teachings within the system of Gore, Jr. and Mighdoll because it would be more efficient to transfer of data from a server system hosting a client system requesting access to the website.

The Examiner is correct in stating that Shieh's client server architecture would be useful in a system of Gore, Jr. and Mighdoll. Applicant respectfully submits that the application of a client server architecture or the addition of the blocking software of Shieh does little, however, to build a system that determines the user's identity, determines the role or roles available to that user and

limits access to documents based on those roles. Applicant respectfully requests reconsideration of claims 1-6 and 35.

Claims 1-6 and 35 also include, among other things, building a document list naming documents available to clients assigned to a particular role. The Examiner states that Gore Jr. does not teach how to build a document list but that Mighdoll teaches a method to retrieve a document from a remote server in response to a client request and to generate a display based on the document.

The Examiner states that the addition of Mighdoll's document retrieving method to Gore, Jr. is sufficient to teach sufficient to teach a system which limits access to documents based on roles assigned to users. There is no explanation of why, however, the mere retrieval of a document would suggest limiting access to documents based on roles and there is no suggestion in any of the cited references to do so. Thus, claims 1-6 and 35 are allowable.

In addition, Gore Jr. teaches away from the present invention. Applicant's invention, as described in the specification and recited in claims 1-6 and 35, allows the initiating action to be a request for a document from an external network. Gore Jr. does not allow a connection to be initiated externally. (See abstract, col.1, lines 57-61 and col. 4, lines 10-41). For instance, Gore Jr. requires a connection to be set up between an internal and an external server, initiated by the internal server, before a customer/user can successfully request a transaction. (See col. 4, lines 10-12). Thus, Gore Jr. teaches away from the present invention, as described in the specification and recited in claims 1-6 and 35, making claims 1-6 and 35 allowable.

Claims 2-6 distinguish over Gore Jr, Mighdoll and Shieh in other ways as well. For instance, claims 2 and 5 include each document having a unique URL and the document list including the URL of each document. Applicant respectfully submits that since the cited patents do not show or teach building a document list they do not teach each document having a unique URL and including the URL in the document list.

With respect to claims 3 and 6, the Examiner states that displaying a document list using a tree structure in a Graphical User Interface (GUI) would be obvious to one of skill in the art. Since all the elements of claims 3 and 6 (in particular, using a tree structure in a GUI to display a

document list) are not found in the prior art, Applicant assumes that the Examiner is taking Official Notice of the missing elements and functions. Applicant respectfully traverses the assertion of Official Notice and submits that using a tree structure in a GUI to display a document list is not well known in the art. Applicant requests that the Examiner cite patents and references in support of his position.

II) The rejection of claims 7-9, 13, 14, 18, 20, 23, 25, 27, 36 and 37 under 35 U.S.C. § 103 as being unpatentable over on Gore

Claims 7-9, 13, 14, 18, 20, 23, 25, 27, 36 and 37 were rejected under 35 U.S.C. 103(a) as being unpatentable over Gore, Jr. et al. (U.S. Patent No. 5,826,029).

Claims 7-10 and 18-22 include, among other things, a document control server with a go list processor. The go list processor is used to determine if a user has access to a document. Applicant is unable to find in any of the cited portions of the Gore Jr. patent a teaching of a go list processor used to determine if a user has access to a document, as recited in the claims.

Further, claims 7-10 and 18-22 include "a document processor for reading the first document from the document server, cleaning the first document and forwarding a clean version of said first document to the user." The Examiner states that, although "Gore, Jr. does not expressly teach the use of *cleaning the first document*...., it would have been obvious to a person of ordinary skill in the art at the time of the invention to employ a document cleaning method within the system of Gore, Jr. because it would allow a client to access an internal database for clean document without violating security firewalls." For support, the Examiner turns to col. 1, lines 46-52 of Gore, where Gore states:

> Accordingly, a computer implemented method, uniquely programmed computer system, and article of manufacture embodying computer readable program means allow a customer on an external network to initiate an authorized business transaction utilizing internal business resources on an internal network without violating security firewalls.

As noted in the response to the first Office Action, the Examiner must be taking Official Notice of the need for, and of a particular implementation for, a "cleaning" function in the context of a

document retrieval system. Applicant traversed the Examiner's assertion of Official Notice in the response to the first Office Action and requested that the Examiner cite some reference in support of his position. The Examiner has not done so.

Applicant once again submits that cleaning a document to remove or replace embedded links is not well known in the art and one again requests that the Examiner cite some reference in support of his position.

Further, Gore Jr. teaches away from the present invention. Applicant's invention, as described in the specification and recited in claims 7-10 and 18-22, allows the initiating action to be a request for a document from an external interface. Gore Jr. does not allow a connection to be initiated externally. (See abstract, col.1, lines 57-61 and col. 4, lines 10-41). Specifically, Gore Jr. requires a connection be set up between an internal and an external server, initiated by the internal server, before a customer/user can successfully request a transaction. (See col. 4, lines 10-12). Thus, Gore Jr. teaches away from the present invention, as recited in claim 7-10 and 18-22, making claims 7-10 and 18-22 allowable.

Claims 8-10 and 18-22 distinguish over the cited patents in other ways as well. For instance, claim 9 includes a document processor acting as a proxy to hide the mechanics of the access to the first document. The Examiner stated that Gore Jr. teaches a firewall connected to hide access to the first document. Applicant respectfully disagrees. Although a firewall may act as a proxy, there is no teaching in Gore of the use of a proxy to hide access to a document within a system such as is described by Applicant.

In fact, Gore does not even include a document control server. Therefore, he cannot teach how a document control server can communicate with a document server across a firewall as claimed in claims 8 and 18-20, how a telephone connection can be made to such a control server as claimed in claim 10, how documents can be cleaned as claimed in claim 21 or how access can be authorized as claimed in claim 22.

Claims 23-29, 36 and 37 also distinguish over Gore. For instance, claims 23-29, 36 and 36 include, among other things, defining documents accessible to the users. The Examiner has stated that Gore Jr. does not teach or suggest defining documents accessible to the users. (See

Paper no. 10, page 4). Since Gore Jr. does not teach or suggest defining documents accessible to the users and claims 23-29, 36 and 37 include the limitation of defining documents accessible to the users, claims 23-29, 36 and 37 are allowable.

In addition, claims 23-29, 36 and 37 include the limitation of cleaning a document of embedded links. The Examiner stated that cleaning a document would be obvious to one skill in the art to "allow a client to access an internal database for clean document without violating security firewalls." Since all the elements of claims 23-29, 36 and 37 (e.g., the limitation of cleaning the document of embedded links) are not found in the prior art, Applicant assumes that the Examiner is taking Official Notice of the missing elements and functions. Applicant respectfully traverses the assertion of Official Notice and submits that cleaning a document to remove or modify embedded links is not well known in the art of the present invention. Applicant requests that the Examiner cite patents and references in support of his position.

Further, Gore Jr. teaches away from the present invention. Applicant teaches, and claims in claims 23-29, 36 and 37, that a request for a document will be initiated from an external network. Gore Jr. does not allow a connection to be initiated externally. (See abstract, col.1, lines 57-61 and col. 4, lines 10-41). Specifically, Gore Jr. requires a connection be set up between an internal and an external server, initiated by the internal server, before a customer/user can successfully request a transaction. (See col. 4, lines 10-12). Thus, Gore Jr. teaches away from the present invention recited in claims 23-29, 36 and 37.

Claims 24-29 and 37 distinguish over Gore Jr. in other ways as well. For instance, claims 25 and 26 include assigning one or more roles to users and limiting access to documents as a function or the assigned roles. The Examiner states that this feature is described in Gore Jr. at col. 4 lines 10-26, col. 7 lines 30-67 and col. 8 lines 1-63. Applicant is unable to find in any of the cited portions of Gore Jr. a teaching or suggestion of assigning roles to users and limiting access to documents as a function of the assigned roles.

Further, claim 24 includes the limitation that each document has a unique URL and that the document list includes the URL of each document. Applicant respectfully submits that since

the cited patents do not show or teach building a document list they do not teach each document having a unique URL and including the URL in the document list.

Further still, claims 28, 29 and 37 include limitations including ways of cleaning a document of embedded links. Since the cited patents to do not teach or suggest cleaning a document of embedded links they cannot teach or suggest the ways of cleaning a document in claims 28, 29 and 37.

Therefore, neither Gore, Jr., nor any of the other references, suggest or teach all of the elements of the claimed invention. Applicant respectfully requests the withdrawal of the rejection of claims 7-9, 13, 14, 18, 20, 23, 25, 27, 36 and 37.

III) The rejection of claims 11, 12, 15 and 16 under 35 U.S.C. § 103 based on Gore and Mighdoll

Claims 11, 12, 15 and 16 were rejected under 35 U.S.C. 103(a) as being unpatentable over Gore, Jr. et al. (U.S. Patent No. 5,826,029) in view of Mighdoll et al. (U.S. Patent No. 5,918,013). The rejection of claim 11 has been placed under this section since the Examiner's rejection of claim 11 was based solely on the combination of Gore and Mighdoll.

Claims 11-17 include, among other things, a data owner interface for building a list of available documents. The Examiner admits that Gore Jr. does not show this limitation. The Examiner states, however, that Mighdoll teaches displaying available documents within a graphical user interface and accessing a control server to obtain a current version of the document list. a method to receive a document from a remote server in response to a client request. The Examiner further states that the document includes data to be used by the client in generating a display. However, the cited sections of Mighdoll do not teach or suggest a data owner interface for building a list of available documents, as described in the specification and recited in the claims. Specifically, the cited portions of Mighdoll only show how to retrieve a document and show nothing of an interface for building a document list of available documents. Thus, Mighdoll does not show or teach the building of a list of documents available through a data owner interface, as described in the specification and recited in the claims.

In addition, claims 11-17 include a go list processor for determining, based on a document list, if a user has access to a document. Applicant is unable to find in any of the cited references a teaching of a go list processor for determining, based on a document list, if a user has access to a document, as that processor is described in the specification and recited in the claims.

Further, Gore Jr. teaches away from the present invention. Applicant's invention, as described in the specification and recited in claims 11-17, allows the initiating action to be a request for a document from an external interface. Gore Jr. does not allow a connection to be initiated externally. (See abstract, col.1, lines 57-61 and col. 4, lines 10-41). Specifically, Gore Jr. requires a connection be set up between an internal and an external server, initiated by the internal server, before a customer/user can successfully request a transaction. (See col. 4, lines 10-12). Thus, Gore Jr. teaches away from the present invention, as described in the specification and recited in claims 11-17, making claims 11-17 allowable.

Claims 12, 15 and 16 distinguish over Gore Jr, and Mighdoll in other ways as well. For instance, claims 12 and 15 include a GUI requiring the document server to obtain a current version of the document list. Applicant respectfully submits that since the cited patents do not show or teach a document list they do not show or teach a GUI requiring the document server to obtain a current version of the document list.

IV) The rejection of claims 10, 17, 19, 30 and 32 under 35 U.S.C. § 103 based on Gore and Dustan

Claims 10, 17, 19, 30 and 32 were rejected under 35 USC §103(a) as being unpatentable over Gore, Jr. (US 5,826,029) in view of Dustan et al. (US 5,844,312).

Claims 10 and 19 are dependent on claim 7 and, therefore, incorporate the limitations of claim 7.

Claims 7, 10 and 19 include, among other things, a document control server with a go list processor. The go list processor is used to determine if a user has access to a document. Applicant is unable to find in any of the cited portions of the Gore Jr. or Dustan patents a

teaching of a go list processor used to determine if a user has access to a document, as recited in the claim.

Further, claims 7, 10 and 19 include "a document processor for reading the first document from the document server, cleaning the first document and forwarding a clean version of said first document to the user." The Examiner states that, although "Gore, Jr. does not expressly teach the use of *cleaning the first document*....., it would have been obvious to a person of ordinary skill in the art at the time of the invention to employ a document cleaning method within the system of Gore, Jr. because it would allow a client to access an internal database for clean document without violating security firewalls." As noted above, for support the Examiner turns to col. 1, lines 46-52 of Gore, where Gore states:

> Accordingly, a computer implemented method, uniquely programmed computer system, and article of manufacture embodying computer readable program means allow a customer on an external network to initiate an authorized business transaction utilizing internal business resources on an internal network without violating security firewalls.

As noted in the response to the first Office Action, the Examiner must be taking Official Notice of the need for, and of a particular implementation for, a "cleaning" function in the context of a document retrieval system. As noted above, Applicant traversed the Examiner's assertion of Official Notice in the response to the first Office Action and requested that the Examiner cite some reference in support of his position. The Examiner has not done so.

Applicant once again submits that cleaning a document to remove or replace embedded links is not well known in the art and one again requests that the Examiner cite some reference in support of his position.

Further, Gore Jr. teaches away from the present invention. Applicant's invention, as described in the specification and recited in claims 7, 10 and 19, allows the initiating action to be a request for a document from an external interface. Gore Jr. does not allow a connection to be initiated externally. (See abstract, col.1, lines 57-61 and col. 4, lines 10-41). Specifically, Gore Jr. requires a connection be set up between an internal and an external server, initiated by the internal server, before a customer/user can successfully request a transaction. (See col. 4, lines

10-12). Thus, Gore Jr. teaches away from the present invention, as recited in claim 7-10 and 18-22, making claims 7-10 and 18-22 allowable.

The Examiner rejected claim 17 as being unpatentable over Gore in view of Dustan. Claim 17 is dependent on claim 11; claim 11 was rejected as unpatentable over Gore, Jr. et al. (U.S. Patent No. 5,826,029) in view of Mighdoll et al. (U.S. Patent No. 5,918,013).

In the rejection of claim 11 above, the Examiner stated that "Gore, Jr. does not expressly teach the limitation of 'building a document list of available document.'" *Office Action mailed June 1, 2000*, p. 6, lines 1-2. The Examiner does not explain how Dustan provides such a teaching.

In addition, as noted above, claims 11-17 include, among other things, a data owner interface for building a list of available documents. The Examiner states above that Mighdoll teaches displaying available documents within a graphical user interface and accessing a control server to obtain a current version of the document list. a method to receive a document from a remote server in response to a client request. The Examiner further states that the document includes data to be used by the client in generating a display. However, the cited sections of Mighdoll do not teach or suggest a data owner interface for building a list of available documents, as described in the specification and recited in the claims. Specifically, the cited portions of Mighdoll only show how to retrieve a document and show nothing of an interface for building a document list of available documents. Thus, Mighdoll does not show or teach the building of a list of documents available through a data owner interface, as described in the specification and recited in the claims.

In addition, claims 11-17 include a go list processor for determining, based on a document list, if a user has access to a document. Applicant is unable to find in any of the cited references a teaching of a go list processor for determining, based on a document list, if a user has access to a document, as that processor is described in the specification and recited in the claims.

Claims 30-34 include, among other things, assigning roles to users where the roles can be different and users may have more than one role. Applicant is unable to find in either the Gore

Jr. or Dustan references any teaching of assigning users to one or more roles, as described in the specification and recited in the claims.

The Examiner stated that Gore Jr. shows the role of a client. *Office Action mailed June 1, 2000*, p. 16, lines 5-9. Applicant respectfully submits that assigning roles to clients, as recited in the claims, is different from the role of a client in Gore Jr. Applicant's use of the term "client" is not to be confused with the use of the term "client system" in a client server architecture such as described by Gore or Shieh. Instead, Applicant's clients are the system users (see, e.g., p. 6, line 18 through p. 7, line 15).

In addition, claims 30-34 include defining documents accessible to users where access can be limited based on the assigned roles. The Examiner relied on Gore Jr., Figs. 3 & 4, col. 1 lines 53-67, col. 2 lines 1-8, col. 3 lines 14-62, col. 4 lines 1-62, col. 7 lines 30-67 and col. 8 lines 1-67. However, Applicant is unable to find in any of the cited portions of Gore Jr. a teaching or suggestion of defining documents accessible to users and limiting access to documents as a function of the assigned roles.

Furthermore, claim 32 includes the limitation of cleaning a document of embedded links. Since the cited patents to do not teach or suggest cleaning a document of embedded links they cannot teach or suggest the ways of cleaning a document in claims 32.

Gore Jr. also teaches away from the present invention. In the Applicant's invention, as described in the specification and recited in claims 11-17, 19, and 30-34, the initiating action is a request for a document from an external network. Gore Jr. does not allow a connection to be initiated externally. (See abstract, col.1, lines 57-61 and col. 4, lines 10-41). Specifically, Gore Jr. requires a connection be set up between an internal and an external server, initiated by the internal server, before a customer/user can successfully request a transaction. (See col. 4, lines 10-12). Thus, Gore Jr. teaches away from the present invention, as recited in claims 11-17, 19, and 30-34, making claims 11-17, 19, and 30-34 allowable.

V) The rejection of claim 31 under 35 U.S.C. § 103 based on Gore, Dustan and Izawa

Claim 31 is dependent on claim 30. As noted above, claims 30-34 include, among other things, assigning roles to users where the roles can be different and users may have more than one role. The Examiner stated that Gore Jr. shows the role of a client. As noted above, Applicant respectfully submits that assigning roles to clients, as recited in the claims, is different from the role of a client in Gore Jr. Specifically, Applicant is unable to find in either the Gore Jr. or Dustan references any teaching of assigning users to one or more roles, as described in the specification and recited in the claims.

In addition, claims 30-34 include defining documents accessible to users where access can be limited based on the assigned roles. The Examiner relied on Gore Jr., Figs. 3 & 4, col. 1 lines 53-67, col. 2 lines 1-8, col. 3 lines 14-62, col. 4 lines 1-62, col. 7 lines 30-67 and col. 8 lines 1-67. However, Applicant is unable to find in any of the cited portions of Gore Jr. a teaching or suggestion of defining documents accessible to users and limiting access to documents as a function of the assigned roles.

Furthermore, claim 31 includes the limitation of accessing a document list listing the URL of each available document. As noted above, the Examiner stated that "Gore, Jr. does not expressly teach the limitation of 'building a document list of available document.'" *Office Action mailed June 1, 2000*, p. 6, lines 1-2. The Examiner does not explain how Dustan or Izawa provides such a teaching.

In addition, claim 31 includes the limitation that each document has a unique URL and that the document list includes the URL of each available document. Applicant respectfully submits that since the cited patents do not show or teach a document list, as described in the specification and recited in claim 31, they do not teach each document having a unique URL and accessing the URL in the document list.

VI) The rejection of claims 33 and 34 under 35 U.S.C. § 103 based on Gore, Dustan and Iyengar

Claims 33 and 34 were rejected under 35 USC §103(a) as being unpatentable over Gore, Jr. (US 5,826,029) in view of Dustan and Iyengar (US 5,961,601).

Claims 33 and 34 are dependent on claim 32. As noted above, claim 32 includes cleaning documents of embedded links. The Examiner stated that cleaning a document of embedded links would be obvious to one skill in the art. Since all the elements of claim 32 (in particular, cleaning the document of embedded links) are not found in the prior art, Applicant assumes that the Examiner is taking Official Notice of the missing elements and functions. Applicant respectfully traverses the assertion of Official Notice and submits that cleaning a document of embedded links is not well known in the art. Applicant requests that the Examiner cite patents and references in support of his position.

Further, claims 33-34 include limitations directed to ways of cleaning a document of embedded links. Since the cited patents to do not teach or suggest cleaning a document of embedded links they do not teach or suggest the ways of cleaning a document as claimed in claims 33-34.

Therefore, Gore, Jr., Dustan, and Iyengar, alone or combined, do not teach or suggest all of the elements of the claimed invention, and in fact teach away from the present invention. Applicant respectfully requests the withdrawal of the rejection of claims 33 and 34.

VII) The rejection of claims 24 and 26 under 35 U.S.C. §103 based on Gore and Izawa

Claims 24 and 26 were rejected under 35 USC §103(a) as being unpatentable over Gore, Jr. (US 5,826,029) in view of Izawa et al. (US 5,179,658).

Applicant respectfully traverses the rejection of claims 24 and 26. It is respectfully submitted that Gore, Jr. and Izawa, alone or combined, do not teach or suggest all of the elements of the claimed invention.

Claims 23-29, 36 and 37 distinguish over Gore, either alone or in combination with Izawa. For instance, claims 23-29, 36 and 37 include, among other things, defining documents accessible to the users. The Examiner has stated that Gore Jr. does not teach or suggest defining documents accessible to the users. (See Paper no. 10, page 4). Since Gore Jr. does not teach or suggest defining documents accessible to the users and claims 23-29, 36 and 37 include the limitation of defining documents accessible to the users, claims 23-29, 36 and 37 are allowable.

As noted above, claims 23-29 include the limitation of cleaning a document of embedded links. The Examiner stated that cleaning a document would be obvious to one skill in the art to "allow a client to access an internal database for clean document without violating security firewalls." Since all the elements of claims 23-29 (e.g., the limitation of cleaning the document of embedded links) are not found in the prior art, Applicant assumes that the Examiner is taking Official Notice of the missing elements and functions. Applicant respectfully traverses the assertion of Official Notice and submits that cleaning a document to remove or modify embedded links is not well known in the art of the present invention. Applicant requests that the Examiner cite patents and references in support of his position.

Further, Gore Jr. teaches away from the present invention. Applicant teaches, and claims in claims 23-29, that a request for a document will be initiated from an external network. Gore Jr. does not allow a connection to be initiated externally. (See abstract, col.1, lines 57-61 and col. 4, lines 10-41). Specifically, Gore Jr. requires a connection be set up between an internal and an external server, initiated by the internal server, before a customer/user can successfully request a transaction. (See col. 4, lines 10-12). Thus, Gore Jr. teaches away from the present invention recited in claims 23-29.

Claims 24 and 26 distinguish over Gore Jr. in other ways as well. For instance, claims 24 and 26 include the limitation that each document has a unique URL and that the document list includes the URL of each document. Applicant respectfully submits that since the cited patents do not show or teach building a document list they do not teach each document having a unique URL and including the URL in the document list. Furthermore, claim 26 includes the limitation of determining whether a user requesting a document is in the group of users who have

permission to access the document (i.e., the user has a role which allows him or her to access the document). As noted above, there is no such teaching in either Gore or Izawa.

VIII) The rejection of claims 21, 22, 28 and 29 under 35 U.S.C. § 103 based on Gore and Iyengar

Claims 21, 22, 28 and 29 were rejected under 35 USC §103(a) as being unpatentable over Gore, Jr. (US 5,826,029) in view of Iyengar (US 5,961,601).

Claims 21 and 22 are dependent on claim 7 and, therefore, incorporate the limitations of claim 7.

Claims 7, 21 and 22 include, among other things, a document control server with a go list processor. The go list processor is used to determine if a user has access to a document. Applicant is unable to find in any of the cited portions of the Gore Jr. or Iyengar patents a teaching of a go list processor used to determine if a user has access to a document, as recited in the claim.

Further, claims 7, 21 and 22 include "a document processor for reading the first document from the document server, cleaning the first document and forwarding a clean version of said first document to the user." The Examiner states that, although "Gore, Jr. does not expressly teach the use of *cleaning the first document*...., it would have been obvious to a person of ordinary skill in the art at the time of the invention to employ a document cleaning method within the system of Gore, Jr. because it would allow a client to access an internal database for clean document without violating security firewalls." As noted above, for support the Examiner turns to col. 1, lines 46-52 of Gore, where Gore states:

> Accordingly, a computer implemented method, uniquely programmed computer system, and article of manufacture embodying computer readable program means allow a customer on an external network to initiate an authorized business transaction utilizing internal business resources on an internal network without violating security firewalls.

As noted in the response to the first Office Action, the Examiner must be taking Official Notice of the need for, and of a particular implementation for, a "cleaning" function in the context of a

document retrieval system. As noted above, Applicant traversed the Examiner's assertion of Official Notice in the response to the first Office Action and requested that the Examiner cite some reference in support of his position. The Examiner has not done so.

Applicant once again submits that cleaning a document to remove or replace embedded links is not well known in the art and one again requests that the Examiner cite some reference in support of his position.

Claims 23-29, 36 and 37 distinguish over Gore, either alone or in combination with Izawa. For instance, claims 23-29, 36 and 37 include, among other things, defining documents accessible to the users. The Examiner has stated that Gore Jr. does not teach or suggest defining documents accessible to the users. (See Paper no. 10, page 4). Since Gore Jr. does not teach or suggest defining documents accessible to the users and claims 23-29, 36 and 37 include the limitation of defining documents accessible to the users, claims 23-29, 36 and 37 are allowable.

As noted above, claims 23-29 include the limitation of cleaning a document of embedded links. The Examiner stated that cleaning a document would be obvious to one skill in the art to "allow a client to access an internal database for clean document without violating security firewalls." Since all the elements of claims 23-29 (e.g., the limitation of cleaning the document of embedded links) are not found in the prior art, Applicant assumes that the Examiner is taking Official Notice of the missing elements and functions. Applicant respectfully traverses the assertion of Official Notice and submits that cleaning a document to remove or modify embedded links is not well known in the art of the present invention. Applicant requests that the Examiner cite patents and references in support of his position.

In addition, claims 28 and 29 include the limitation of looking for a server path link or an absolute path link, respectively, and replacing the links with links to an alias or a different link if needed. Applicant respectfully submits that the modified HTML in Iyengar, which is cited by the Examiner, does not teach or suggest looking for a server path link or for an absolute path link. Nor does it teach replacing the link with a link to an alias or different link if needed. Specifically, Iyengar only describes modifying a link to preserve state variables and is not changing the link to an alias or a different link. Further, Iyengar does not show anything that

resembles searching for either a server path link or an absolute path link, as described in the specification and recited in the claims.

Further, Gore Jr. teaches away from the present invention. Applicant teaches, and claims in claims 7, 21, 22 and 23-29, that a request for a document will be initiated from an external network. Gore Jr. does not allow a connection to be initiated externally. (See abstract, col.1, lines 57-61 and col. 4, lines 10-41). Specifically, Gore Jr. requires a connection be set up between an internal and an external server, initiated by the internal server, before a customer/user can successfully request a transaction. (See col. 4, lines 10-12). Thus, Gore Jr. teaches away from the present invention recited in claims 7 and 21-29.

CONCLUSION

Applicant respectfully requests reconsideration of claims 1-37. Applicant submits that the claims are in condition for allowance and notification to that effect is earnestly requested. The Examiner is invited to telephone Applicant's attorney (612-373-6909) to facilitate prosecution of this application.

If necessary, please charge any additional fees or credit overpayment to Deposit Account No. 19-0743.

Respectfully submitted,

RICHARD R. VIETS ET AL.

By their Representatives,

SCHWEGMAN, LUNDBERG, WOESSNER & KLUTH, P.A.
P.O. Box 2938
Minneapolis, MN 55402
(612) 373-6909

Date September 19, 2000 By Thomas F. Brennan

Thomas F. Brennan
Reg. No. 35,075

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to Commissioner of Patents, Washington, D.C. 20231 on September 19, 2000.

THOMAS F. BRENNAN
Name

Thomas F. Brennan
Signature

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/495,157	01/31/2000	Thomas D. Ashoff	NAI1P075/99.039.01	4471

7590 09/10/2003

Schwegman, Lundberg, Woessner & Kluth, P.A.
P.O. Box 2938
Minneapolis, MN 55402

EXAMINER	
MASHAAL, ALI M	

ART UNIT	PAPER NUMBER
2133	10

DATE MAILED: 09/10/2003

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 07-01)

Office Action Summary	Application No.	Applicant(s)	
	09/495,157	ASHOFF ET AL.	
	Examiner	Art Unit	
	Ali M. Mashaal	2133	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *31 January 2000* .

2a)☐ This action is FINAL. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-17* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-17* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *31 January 2000* is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

11)☐ The proposed drawing correction filed on _____ is: a)☐ approved b)☐ disapproved by the Examiner.

If approved, corrected drawings are required in reply to this Office action.

12)☒ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. §§ 119 and 120

13)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____ .

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).

a) ☐ The translation of the foreign language provisional application has been received.

15)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449) Paper No(s) *3* .
4) ☐ Interview Summary (PTO-413) Paper No(s). _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other:

DETAILED ACTION

Oath/Declaration

1. The oath or declaration is defective. A new oath or declaration in compliance with 37 CFR 1.67(a) identifying this application by the application number and filing date is required. See MPEP §§ 602.01 and 602.02.

The oath or declaration is defective because:
The "[ ] is attached to option is marked, when in fact the declaration was filed after the application, and therefore could not have been attached.

Specification

2. The disclosure is objected to because it contains an embedded hyperlink and/or other form of browser-executable code. For example on page 4, lines 11 and 12, "http://www.stanford.edu/~hodges/talks/mactivity.ldap.97/index2.html". Applicant is required to delete **all** embedded hyperlinks and/or other forms of browser-executable code. See MPEP § 608.01.

Drawings

3. Figures 1, 4, and 5 should be designated by a legend such as --Prior Art-- because only that which is old is illustrated. See MPEP § 608.02(g). A proposed drawing correction or corrected drawings are required in reply to the Office action to avoid abandonment of the application. The objection to the drawings will not be held in abeyance.

Claim Rejections - 35 USC § 101

4. 35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Claims 1-7 are rejected as being directed to non-statutory subject matter.

Claim 1 is directed to a system for authorizing client access. However, the system comprises a directory, firewall, and authorization filter. As mentioned on page 2 of the Specification, a firewall can be software alone, see lines 10-12. The directory, asserted in figure 4, is a collection of data. The filter is the criteria used for authentication and is intangible; see page 11, lines 7-10, and page 14, lines 15 and 16. Each of the components in claim 1 is disclosed as being implemented as software alone with no tangible elements.

Claims 2-7 each further limit claim 1 by adding an intangible feature as disclosed below.

Claim 2 limits claim 1, only to specify an LDAP directory and therefore is rejected over claim 1.

Claim 3 limits claim 1, only to specify the filter using a GUI and therefore is rejected over claim 1.

Claims 4 and 5 limit claim 1, only to specify the implementation of a per-user and per-service scheme respectively, and therefore are rejected over claim 1.

Claim 6 limits claim 1, only to specify SSL as the protocol for communication between the firewall and the directory, and therefore is rejected over claim 1.

Claim 7 limits claim 1, only by specifying multiple directories and therefore is rejected over claim 1.

Claim Rejections - 35 USC § 103

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. Claims 1-17 are rejected under 35 U.S.C. 103(a) as being unpatentable over US patent number 6,131,120 to Reid in view of The Microsoft Computer Dictionary, 1997, further in view of Check Point Account Management Client, Version 1.0, September 1998.

7. In reference to claims 1, 8, and 17, Reid substantially discloses a system comprising at least one directory, column 5, lines 23-31, that can be accessed using a network protocol. Although not explicitly mentioned, the only possible way for the directory to communicate with the various components on the network illustrated in figure 4, is through the use of a network protocol.

Column 5 lines 23-35 and column 6 lines 54-65, discuss the possibility of deleting or omitting a firewall and implementation of the firewall functions into other network elements. However, the examiner first notes that per the Microsoft Computer Dictionary, a firewall is defined as a security system intended to protect an organization's network against external threats; a firewall prevents computers in the organization's network from communicating directly with computers external to the network and vice versa. In Reid, the firewall functions are integrated into a router/gateway receiving information (Router Access List) from a directory server, see column 6 lines 17-26 and lines 57-61. Acting as the firewall, the routers/gateways

download the access list, and grant or deny access correspondingly. The examiner asserts that while the reference teaches elimination of a separate firewall, it does not in fact eliminate the firewall altogether but instead combines the functions of a firewall with that of a router/gateway into a single unit.

This firewall is configured to intercept network resource requests, see Reid, column 8, lines 6-11, which outlines that the users are either allowed or denied access by the router/gateway. This means that the router/gateway intercepts the users network requests.

Also, column 8, line 9, says "each user" in reference to users accessing the WAN. This establishes a plurality of users.

As per comparison of the authorization filter to directory entries, the examiner asserts that normal and well-known function of a firewall is to selectively control what client user has access to resources it protects. Thus, some type of authorization filter (i.e. filter relative to authorization information) would have to be executed. In Reid, see column 8 lines 14-21, RAL, or router/gateway access list is sent to each router/gateway and controls who may have access through router/gateway. The examiner asserts that in order for the directory to generate the appropriate access list, each router/gateway must have transmitted its access criteria to the directory. The examiner further asserts that this criteria is an authorization filter and that in order for the directory to send back a correct access list, some comparison must have been made with directory entries and the router/gateway criteria (authorization filter).

As per the filter being generated based on schema that is predefined by the entity, Reid discloses that the access list is all ready set in the master directory and is then downloaded to the router/gateway, column 8, lines 14-21. This asserts that the schema is predefined, and that the filter is generated based on it.

We have established to this point that Reid teaches a system for authorizing client access to a network resource having one or more directories that can be accessed through a network protocol, and a firewall that is configured to intercept network resource requests from a plurality of clients, in which the firewall authorizes the requests of the clients based on one or more entries in the said directory to an authorization filter, wherein the authorization filter is generated based on a directory schema that is predefined by the said entity.

As per the directory being configured to store information concerning an organization's entity, Reid's directory, is configured to store names, workstations, router/gateways, servers, IP addresses, locations and so on, column 5, lines 36-39. Reid's directory does not store information concerning an entity's organization. Check Point Account Management Client (disclosed in applicants IDS) teaches storing an entity's organization in a directory tree, page 2, figure 1-1 LDAP Tree Example. Since this structure can be used to authenticate users, it would have been obvious to one of ordinary skill in the art at the time of the invention to take the analogous storage directory tree of Reid and modify it such that Reid's directory tree stored an entity's organization similar to that if Checkpoint. Examiner also notes that Reid implies that it is not necessarily true that what he chooses to store in the directory is the only thing

that can be stored, refer to column 7 line 61, when Reid says "In the embodiment of this invention, the objects may be individual's names....".

8. In reference to claims 2 and 9 which further limit claims 1 and 8 respectively by specifying the directory as being an LDAP directory, Reid teaches a system analogous to that in claims 1 and 8 as mentioned above, and also teaches the use of LDAP directories. See column 4, lines 7-11, column 6, lines 20-24, and column 8, lines 28-31.

9. In reference to claims 4 and 5, which each depend from claim 1, Reid substantially teaches a system analogous to that in claim 1 as mentioned above, and also teaches that the directory contains objects with associated attributes. Specifically, Reid says that the users, router/gateways, and servers are objects. The examiner asserts that Reid's invention teaches both per-user and per-server authentication since this object-oriented directory is organized such that users, router/gateways, and servers are all objects each of which having attributes that include IP address, password, privileges, and location. See column 6, lines 13-20. Accordingly, Reid substantially suggests that any of the two authentication methods could be used.

10. Claims 6 and 13 are rejected under 35 U.S.C. 103(a) as being unpatentable over Reid as applied to claims 1 and 8 above, and further in view of US patent number 5,657,390 to Elgamal. Reid's invention fails to teach the use of SSL as means to secure the communication between the firewall and the directory. Elgamal's analogous

invention teaches that because the socket layer is widely used in networks, integration of SSL into machines that are connected to the network and receive requests would be facilitated. See column 1, lines 60-63. It would have been obvious to one having ordinary skill in the art at the time the invention was made to take Reid's firewall and use the SSL method taught by Elgamal to communicate between the firewall and the directory. One having ordinary skill in the art at the time the invention was made would have been motivated to do so because Elgamal establishes a need for SSL as a security mechanism between various applications to transfer various data between one another. See column 1, lines 44-54. Furthermore, the nature of the information being transferred between the firewall and the directory in the applicant's invention is private and sensitive.

11. In reference to claims 7 and 14, which further limit claims 1 and 8 respectively by specifying multiple directories being queried, Reid substantially teaches a system analogous to that in claims 1 and 8 as mentioned above, and also teaches the use of multiple directories as described in column 7, lines 58-61, when he refers to distributed directories and a master directory.

12. In reference to claims 15 and 16 which further limit claim 8 by specifying that the request comes from an internal user and an external user respectively, Reid substantially teaches a system analogous to that in claim 8 as mentioned above, and also states that his system handles both internal and external requests. Refer to

column 7, lines 39-44, when Reid says that the security policy is defined whether the user is internal or external to the network.

13. Claims 3 and 10 are rejected under 35 U.S.C. 103(a) as being unpatentable over Reid as applied to claims 1 and 8 above, and further in view of US patent number 5,898,830 to Wesinger. Reid discloses a system that encompasses all the limitations of claims 1 and 8, but fails to teach the use of a GUI interface to specify the authorization filter. Wesinger, in an analogous art, teaches the use of a graphical user point and click web interface for configuring the firewall and specifying configuration parameters. It would have been obvious to one having ordinary skill in the art at the time the invention was made to modify the firewall of Reid to include a GUI interface by which the authorization filter could be specified. One having ordinary skill in the art at the time the invention was made would have been motivated to do so because graphical user interfaces have become highly popular and favored for their ease of use, and because they are cheap and easy to develop.

14. Claims 11 and 12 are rejected under 35 U.S.C. 103(a) as being unpatentable over Reid in view of Wesinger as applied to claims 3 and 10 above, and further in view of Reid as applied to claims 4 and 5 above.

Conclusion

The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

The following patents are cited to further show the state of the art with respect to directory-based access controlled networks:

U.S. Pat. No. 006047322A to Vaid

Pub. No. 20030126468 to Markham

U.S. Pat. No. 006212558B1 to Antur

U.S. Pat. No. 006233688B1 to Montenegro

U.S. Pat. No. 006324648B1 to Grantges

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ali M. Mashaal whose telephone number is 703-305-7854. The examiner can normally be reached on 8:30-5:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Albert Decady can be reached on 703-305-9595. The fax phone number for the organization where this application or proceeding is assigned is (703) 872-9306.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is 703-305-3800.

A M

ALBERT DECADY
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

09/04/03

Notice of References Cited	Application/Control No. 09/495,157	Applicant(s)/Patent Under Reexamination ASHOFF ET AL.	
	Examiner Ali M. Mashaal	Art Unit 2133	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
	A	US-6,131,120	10-2000	Reid, William J.	709/225
	B	US-5,898,830	04-1999	Wesinger et al.	713/201
	C	US-5,657,390	08-1997	Elgamal et al.	713/151
	D	US-6,047,322	04-2000	Vaid et al.	709/224
	E	US-6,212,558 B1	04-2001	Antur et al.	709/221
	F	US-2003/0126468 A1	07-2003	Markham, Thomas R.	713/201
	G	US-6,233,688 B1	05-2001	Montenegro, Gabriel	713/201
	H	US-6,324,648 B1	11-2001	Grantges, Jr., David R.	713/201
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	Microsoft Corporation, Microsoft Computer Dictionary, Microsoft Press, Third edition, page 197.
	V	
	W	
	X	

A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
ates in MM-YYYY format are publication dates. Classifications may be US or foreign.

2136
2133

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant: Thomas D. Ashoff et al.

Title: System, Method and Computer Program Product for Authenticating Users Using a Lightweight Directory Access Protocol (LDAP) Directory Server

#11/A
1-23-04
B.O.H

Docket No.: 105.201US1
Filed: January 31, 2000
Examiner: Ali M Mashaal
Customer No.: 21186

OIPE JC1 JAN 16 2004 PATENT & TRADEMARK OFFICE

Serial No.: 09/495157
Due Date: January 10, 2004
Group Art Unit: ~~2133~~ 2136

RECEIVED
JAN 20 2004
Technology Center 2100

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

We are transmitting herewith the following attached items (as indicated with an "X"):

X A return postcard.
X An Amendment and Response (11 Pages).
X Petition for Extension of Time (1 pg.)
X Please charge the amount of $55.00 to Deposit Account No. 19-0743 to cover the Extension of Time Fee.
X Corrected Drawings (1 pg.).

If not provided for in a separate paper filed herewith, Please consider this a PETITION FOR EXTENSION OF TIME for sufficient number of months to enter these papers and please charge any additional fees or credit overpayment to Deposit Account No. 19-0743.

SCHWEGMAN, LUNDBERG, WOESSNER & KLUTH, P.A.
Customer Number 21186

By: Thomas F. Brennan
Atty: Thomas F. Brennan
Reg. No. 35,075

CERTIFICATE UNDER 37 CFR 1.8: The undersigned hereby certifies that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail, in an envelope addressed to: Commissioner for Patents, P.O.Box 1450, Alexandria, VA 22313-1450, on this 12 day of January, 2004.

Gina Uphus
Name

Gina Uphus
Signature

SCHWEGMAN, LUNDBERG, WOESSNER & KLUTH, P.A. Customer Number 21186
(GENERAL)

01/26/2004 BHILLIAR 00000001 190743 09495157
01 FC:2251 55.00 DA

IN THE DRAWINGS

Corrected drawings are supplied herewith.

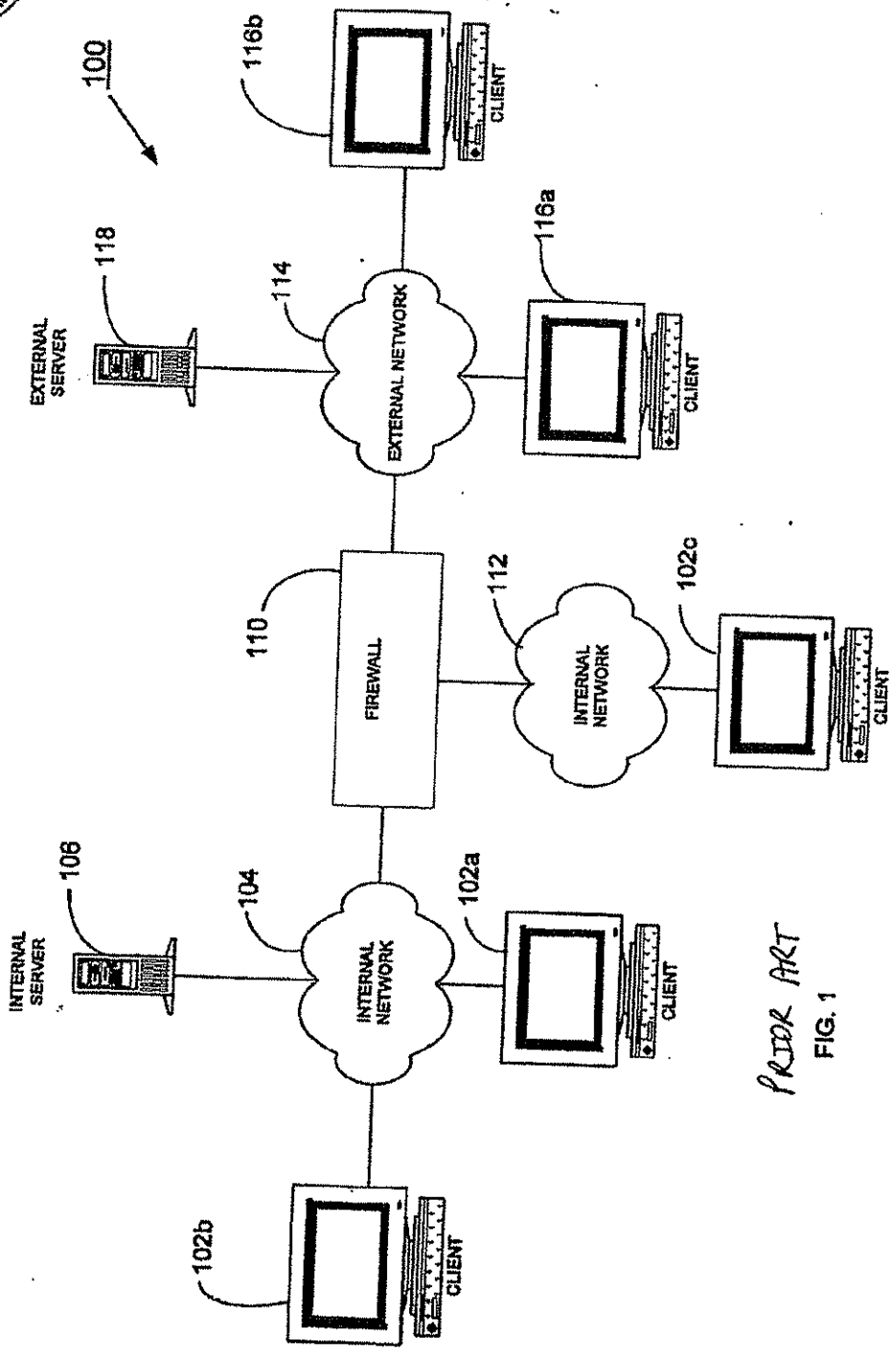
Enclosed is a copy of Figure 1 of the drawings showing the proposed amendment adding the label "PRIOR ART" to Figure 1 in red ink. Figures 4 and 5 illustrate aspects of the present invention; it would not be proper to add the "PRIOR ART" label to these figures.

IN THE SPECIFICATION

The paragraph beginning at page 4, line 4 is amended as follows:

A1

Today, a new protocol, lightweight directory access protocol (LDAP), is gaining wide acceptance in business. The LDAP standard defines an information model for a directory, a namespace for defining how directory information is referenced and organized, and a network protocol for accessing information in the directory. LDAP can also include an application programming interface (API). The LDAP protocol mandates how client and server computers can communicate with a LDAP directory. However, LDAP does not mandate how data should be stored. ~~LDAP directories are described in greater detail in "Introduction to Directories and the Lightweight Directory Access Protocol," available at http://www.stanford.edu/~hodges/talks/mactivity-ldap.97/index2.html.~~ More and more companies today use an LDAP directory server to store a database of employees. The LDAP directory generally can store an employee name, phone number, address and other information about the employee, and a password for modifying the employee's information.



PRIOR ART
FIG. 1

IN THE CLAIMS

Please amend the claims as follows:

1. (Currently Amended) A system for authorizing client access to a network resource, comprising:

a server having at least one directory that can be accessed using a network protocol, said at least one directory being configured to store information concerning an entity's organization; and

a firewall that is configured to intercept network resource requests from a plurality of client users, said firewall being operative to authorize a network resource request based upon a comparison of the contents of at least part of one or more entries in said at least one directory to an authorization filter, wherein said authorization filter is generated based on a directory schema that is predefined by said entity.

2. (Original) The system of claim 1, wherein said at least one directory is a lightweight directory access protocol directory.

A2

3. (Original) The system of claim 1, wherein said authorization filter is specified using a graphical user interface.

4. (Original) The system of claim 1, wherein said authorization filter implements a per-user authentication scheme.

5. (Original) The system of claim 1, wherein said authorization filter implements a per-service authentication scheme.

6. (Original) The system of claim 1, wherein said firewall and said directory communicate using secure socket layer communication.

7. (Original) The system of claim 1, wherein said firewall is configured to query multiple directories.

8. (Original) An authentication method at a firewall, comprising the steps of:

(a) receiving a network resource request from a client user;

(b) querying, using a network protocol, at least one directory that is configured to store information concerning an entity's organization, wherein said query is based upon an authorization filter that is generated based on a directory schema that is predefined by said entity;

(c) determining, based on the results of said query, whether the contents of at least part of one or more entries in said at least one directory satisfy said authorization filter; and

(d) permitting said network resource request through said firewall if said authorization filter is satisfied.

A2

9. (Original) The method of claim 8, wherein step (b) comprises the step of querying said at least one directory using a lightweight directory access protocol.

10. (Original) The method of claim 8, further comprising the step of specifying an authorization filter using a graphical user interface.

11. (Original) The method of claim 10, wherein said specifying step comprises the step of specifying an authorization filter that implements a per-user authentication scheme.

12. (Original) The method of claim 10, wherein said specifying step comprises the step of specifying an authorization filter that implements a per-service authentication scheme.

13. (Original) The method of claim 8, wherein step (b) comprises the step of querying said directory using secure socket layer communication.

14. (Original) The method of claim 8, wherein step (b) comprises the step of querying multiple directories.

15. (Original) The method of claim 8, wherein step (a) comprises the step of receiving a network resource request from a client user at an internal network.

16. (Original) The method of claim 8, wherein step (a) comprises the step of receiving a network resource request from a client user at an external network.

17. (Original) A computer program product for enabling a processor in a computer system to implement an authentication process, said computer program product comprising:

a computer usable medium having computer readable program code embodied in said medium for causing a program to execute on the computer system, said computer readable program code comprising:

first computer readable program code for enabling the computer system to receive a network resource request from a client user;

second computer readable program code for enabling the computer system to query, using a network protocol, at least one directory that is configured to store information concerning an entity's organization, wherein said query is based upon an authorization filter that is generated based on a directory schema that is predefined by said entity;

third computer readable program code for enabling the computer system to determine, based on the results of said query, whether the contents of at least part of one or more entries in said at least one directory satisfy said authorization filter; and

fourth computer readable program code for enabling the computer system to permit said network resource request through said firewall if said authorization filter is satisfied.

REMARKS

Applicant has carefully reviewed and considered the Office Action mailed on September 10, 2003, and the references cited therewith.

Claim 1 was amended. No new claims were added. Claims 1-17 remain pending in this application.

§101 Rejection of the Claims

Claims 1-7 were rejected under 35 USC § 101 as being directed to non-statutory matter. Claim 1 has been amended to include a reference to a server where the directory is stored.

§103 Rejection of the Claims

Claims 1-17 were rejected under 35 USC § 103(a) as being unpatentable over Reid (U.S. Patent No. 6,131,120) in view of "The Microsoft Computer Dictionary, 1997", further in view of "Check Point Account Management Client, Version 1.0, 1998".

Reid describes a network security protocol in which router/gateways are used to control network traffic passing through each router/gateway.

> An enterprise directory residing on a directory server stores the names, workstations, router/gateways, servers, IP addresses locations, passwords, and encryption keys for individuals. Periodically, the directory server downloads to each router/gateway across the WAN router/gateway access lists (RALs), thereby controlling all network access across the WAN. Also periodically, the directory server downloads user control files (UCFs) to servers in the network, thereby controlling all server access across the WAN. This directory-based invention thus provides enhanced network control, and enhanced network security.

Reid, col. 6, lines 2-12. Reid, therefore, controls access to data within a network by limiting traffic through router/gateways (using a RAL at each router/gateway) and by limiting access to files within servers (using a UCF at each server). Both the RAL and the UCF are generated by a directory server and distributed periodically to their respective router/gateways and servers.

In addition, one or more of the servers can provide user authentication. Reid states that authentication at the server level is superior to that at the firewall since "distributed authentication provides greatly enhanced security over a firewall-protected network." Reid, col. 6, lines 63-65.

Applicant teaches that it can be difficult to maintain a directory for computer security at the same time that one is maintaining a directory for other purposes. As noted at p. 4, lines 16-23, maintenance of a firewall authentication database is especially burdensome in companies with a large amount of employee turnover or in companies with a large number of firewalls. Applicant teaches that it can be advantageous to configure the firewall to leverage existing databases, such as an LDAP server storing employee information such as is shown in Fig. 4.

As is described on p. 9, line 1 through p. 10, line 2 and as is shown in Fig. 3, an authorization module 206 within firewall 110 receives a request from a user to access an application or resource on the other side of firewall 110. Authorization module 206 authenticates the user, and then determines whether that user is authorized to have his access request fulfilled by querying a server 106 having an LDAP directory. The entry read from the LDAP directory that is associated with the user is compared to an authorization filter. If one or more attributes of the entry does not satisfy the filter, the user is not authorized to access the requested application or resource and the request fails. If, however, all the attributes of the entry satisfy the filter, the user is authorized to access the requested application or resource and the request is allowed through firewall 110.

Claim 1, as amended, includes a server having at least one directory and a firewall configured to intercept network resource requests from a plurality of client users. The firewall is "operative to authorize a network resource request based upon a comparison of the contents of at least part of one or more entries in said at least one directory to an authorization filter, wherein said authorization filter is generated based on a directory schema that is predefined by said entity."

The Examiner stated that since the router/gateway access list is sent to each router/gateway and controls who may have access through router/gateway,

> The examiner asserts that in order for the directory to generate the appropriate access list, each router/gateway must have transmitted its access criteria to the directory. The examiner further asserts that this criteria is an authorization filter and that in order for the directory to send back a correct access list, some comparison must have been made with directory entries and the router/gateway criteria (authorization filter).

Office Action, p. 5, third full paragraph. Applicant disagrees.

As noted above, Reid makes it clear (col. 6, lines 2-12) that all security information is stored within an enterprise directory and that periodically that directory downloads RALs and UCFs to router/gateways and servers, respectively. "Because the directory knows the location and IP address of each user, and the location and IP address of each router/gateway, a directory application can periodically populate the RAL in each router/gateway on the network using LDAP. Entries in the directory thereby control the entire network and the network router/gateway configuration management is automated." Col. 6, lines 19-25. Therefore, despite the Examiner's statement that "in order for the directory to send back a correct access list, some comparison must have been made with directory entries and the router/gateway criteria (authorization filter)," no such authorization filter is provided in Reid. At least one limitation of claims 1-7 is, therefore, not present in any of the references cited by the Examiner. Reconsideration of claims 1-7 is respectfully requested.

Claim 8 is to an authentication method which determines if a user is authorized to access an application or a resource by applying an authorization filter to an entry associated with the user stored in a directory. Once again, Reid does not describe the application of an authorization filter to an entry read from a directory. At least one limitation of claims 8-16 is, therefore, not present in any of the references cited by the Examiner. Reconsideration of claims 8-16 is respectfully requested.

Claim 17 is a computer program product which includes program code for applying an authorization filter to an entry associated with the user stored in a directory. Once again, Reid does not describe the application of an authorization filter to an entry read from a directory. At least one limitation of claim 17 is, therefore, not present in any of the references cited by the Examiner. Reconsideration of claim 17 is respectfully requested.

Claims 6 and 13 were rejected under 35 USC § 103(a) as being unpatentable over Reid as applied to 1 and 8 above, and further in view of Elgamal (U.S. Patent No. 5,657,390).

Neither Reid nor Elgamel teach the use of an authorization filter as described by Applicant. In addition, Elgamal does not describe the use of a secure socket layer communication to distribute an entry in an LDAP database as taught by Applicant and claimed in claims 6 and 13. Reconsideration of claims 6 and 13 is respectfully requested.

Claims 3 and 10 were rejected under 35 USC § 103(a) as being unpatentable over Reid as applied to claims 1 and 8 above, and further in view of Wesinger (U.S. Patent No. 5,898,830).

Neither Reid nor Wesinger teach the use of an authorization filter as described by Applicant. In addition, Wesinger does not describe the use of GUI to specify how to implement the authorization filer as taught by Applicant and claimed in claims 3 and 10. Reconsideration of claims 3 and 10 is respectfully requested.

Claims 11 and 12 were rejected under 35 USC § 103(a) as being unpatentable over Reid in view of Wesinger as applied to claims 3 and 10 above, and further in view of Reid as applied to claims 4 and 5 above.

Neither Reid nor Wesinger teach the use of an authorization filter as described by Applicant. In addition, Wesinger does not describe the methods used to implement the authorization filer as taught by Applicant and claimed in claims 11 and 12. Reconsideration of claims 11 and 12 is respectfully requested.

Conclusion

Applicant respectfully submits that the claims are in condition for allowance and notification to that effect is earnestly requested. The Examiner is invited to telephone Applicant's attorney at (612) 373-6909 to facilitate prosecution of this application.

If necessary, please charge any additional fees or credit overpayment to Deposit Account No. 19-0743

Respectfully submitted,

Thomas D. Ashoff, et al.

By their Representatives,

SCHWEGMAN, LUNDBERG, WOESSNER & KLUTH, P.A.
P.O. Box 2938
Minneapolis, MN 55402
(612) 373-6909

Date January 12, 2004 By Thomas F. Brennan

Thomas F. Brennan
Reg. No. 35,075

CERTIFICATE UNDER 37 CFR 1.8: The undersigned hereby certifies that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail, in an envelope addressed to: Commissioner of Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on this 12 day of January, 2004.

GINA OPHUS

Name

Gina Ophus

Signature

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/495,157	01/31/2000	Thomas D. Ashoff	NAII P075/99.039.01	4471

7590 02/18/2004

Schwegman, Lundberg, Woessner & Kluth, P.A.
P.O. Box 2938
Minneapolis, MN 55402

EXAMINER	
MASHAAL, ALI M	

ART UNIT	PAPER NUMBER
2136	

DATE MAILED: 02/18/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 10/03)

Office Action Summary	Application No.	Applicant(s)
	09/495,157	ASHOFF ET AL.
	Examiner	Art Unit
	Ali M. Mashaal	2136

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *16 January 2004*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-17* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-17* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1) This action is in response to communication filed 01/16/2004.

2) Claims 1-17 are still under examination.

Response to Arguments

3) Applicant's arguments filed 16 January, 2004 have been fully considered by the office but they are not persuasive for the following reasons:

3.1) Applicant argues that with respect to claims 1, 8, 17 (and all other claims since they depend from these) that the applied prior art (specifically Reid) does not teach an authorization filter, by which the firewall can make a comparison of the content of at least part of one or more entries in the directory. However, the office respectfully disagrees with the applicant's position asserting that Reid does in fact teach such an authorization filter as previously mentioned in the office action filed 10 September, 2003. Reviewing again col. 7, line 45-col. 8 line 21 in detail, Reid discloses a configuration in which each router/gateway contains an updated router access list (RAL) which is distributed by a master directory and based on schema contained therein which may include names, workstations, servers, router/gateways, IP addresses, locations, passwords, and encryption keys. Each RAL as populated by the master directory includes the names and addresses of users and the destinations they are allowed to reach. This means that the router/gateway makes use of schema in the directory which is forwarded to it, namely addresses of devices and where they may be forwarded, and compares the requesting device's address and requested destination to that information

in the router/gateway which was provided by the directory server, in order to determine whether the requesting device should be allowed/denied access. Therefore, the router/gateway clearly contains an authorization filter by which it can make a comparison of the content of at least part of one or more entries in the directory to determines which traffic may be allowed to pass through to a given destination.

3.2) With respect to claims 6 and 13, applicant further argues that neither Reid nor Elgamel teach the use of a SSL communication to distribute an entry in an LDAP database. Again the office respectfully disagrees. Elgamel's invention teaches using SSL between devices on a network for added security. Furthermore, it is well-known in the art to do so.

3.3) With respect to claims 3 and 10, applicant further argues that Wesinger does not teach the use of GUI to specify how to specify the authorization filter. Again the office respectfully disagrees. Wesinger teaches configuration as explained in the previous office action.

3.4) With respect to claims 11 and 12, applicant further argues that Wesinger does not teach the claimed methods used to implement the authorization filter. Again the office respectfully disagrees. Wesinger teaches the methods used to implement the authorization filter as explained in the previous office action.

Claim Rejections - 35 USC § 103

4) The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4.1) Claims 1-17 are still rejected under 35 U.S.C. 103(a) as being unpatentable over US patent number 6,131,120 to Reid in view of The Microsoft Computer Dictionary, 1997, further in view of Check Point Account Management Client, Version 1.0, September 1998.

In reference to claims 1, 8, and 17, Reid substantially discloses a system comprising at least one directory, column 5, lines 23-31, that can be accessed using a network protocol. Although not explicitly mentioned, the only possible way for the directory to communicate with the various components on the network illustrated in figure 4, is through the use of a network protocol.

Column 5 lines 23-35 and column 6 lines 54-65, discuss the possibility of deleting or omitting a firewall and implementation of the firewall functions into other network elements. However, the examiner first notes that per the Microsoft Computer Dictionary, a firewall is defined as a security system intended to protect an organization's network against external threats; a firewall prevents computers in the organization's network from communicating directly with computers external to the

network and vice versa. In Reid, the firewall functions are integrated into a router/gateway receiving information (Router Access List) from a directory server, see column 6 lines 17-26 and lines 57-61. Acting as the firewall, the routers/gateways download the access list, and grant or deny access correspondingly. The examiner asserts that while the reference teaches elimination of a separate firewall, it does not in fact eliminate the firewall altogether but instead combines the functions of a firewall with that of a router/gateway into a single unit.

This firewall is configured to intercept network resource requests, see Reid, column 8, lines 6-11, which outlines that the users are either allowed or denied access by the router/gateway. This means that the router/gateway intercepts the users network requests.

Also, column 8, line 9, says "each user" in reference to users accessing the WAN. This establishes a plurality of users.

As per comparison of the authorization filter to directory entries, the examiner asserts that normal and well-known function of a firewall is to selectively control what client user has access to resources it protects. Thus, some type of authorization filter (i.e. filter relative to authorization information) would have to be executed. In Reid, see column 8 lines 14-21, RAL, or router/gateway access list is sent to each router/gateway and controls who may have access through router/gateway. The examiner asserts that in order for the directory to generate the appropriate access list, each router/gateway must have transmitted its access criteria to the directory. The examiner further asserts that this criterion is an authorization filter and that in order for the directory to send back

a correct access list, some comparison must have been made with directory entries and the router/gateway criteria (authorization filter).

As per the filter being generated based on schema that is predefined by the entity, Reid discloses that the access list is all ready set in the master directory and is then downloaded to the router/gateway, column 8, lines 14-21. This asserts that the schema is predefined, and that the filter is generated based on it.

We have established to this point that Reid teaches a system for authorizing client access to a network resource having one or more directories that can be accessed through a network protocol, and a firewall that is configured to intercept network resource requests from a plurality of clients, in which the firewall authorizes the requests of the clients based on one or more entries in the said directory to an authorization filter, wherein the authorization filter is generated based on a directory schema that is predefined by the said entity.

As per the directory being configured to store information concerning an organization's entity, Reid's directory, is configured to store names, workstations, router/gateways, servers, IP addresses, locations and so on, column 5, lines 36-39. Reid's directory does not store information concerning an entity's organization. Check Point Account Management Client (disclosed in applicants IDS) teaches storing an entity's organization in a directory tree, page 2, figure 1-1 LDAP Tree Example. Since this structure can be used to authenticate users, it would have been obvious to one of ordinary skill in the art at the time of the invention to take the analogous storage directory tree of Reid and modify it such that Reid's directory tree stored an entity's

organization similar to that if Checkpoint. Examiner also notes that Reid implies that it is not necessarily true that what he chooses to store in the directory is the only thing that can be stored, refer to column 7 line 61, when Reid says "In the embodiment of this invention, the objects may be individual's names....".

In reference to claims 2 and 9 which further limit claims 1 and 8 respectively by specifying the directory as being an LDAP directory, Reid teaches a system analogous to that in claims 1 and 8 as mentioned above, and also teaches the use of LDAP directories. See column 4, lines 7-11, column 6, lines 20-24, and column 8, lines 28-31.

In reference to claims 4 and 5, which each depend from claim 1, Reid substantially teaches a system analogous to that in claim 1 as mentioned above, and also teaches that the directory contains objects with associated attributes. Specifically, Reid says that the users, router/gateways, and servers are objects. The examiner asserts that Reid's invention teaches both per-user and per-server authentication since this object-oriented directory is organized such that users, router/gateways, and servers are all objects each of which having attributes that include IP address, password, privileges, and location. See column 6, lines 13-20. Accordingly, Reid substantially suggests that any of the two authentication methods could be used.

4.2) Claims 6 and 13 are still rejected under 35 U.S.C. 103(a) as being unpatentable over Reid as applied to claims 1 and 8 above, and further in view of US patent number 5,657,390 to Elgamal. Reid's invention fails to teach the use of SSL as means to

secure the communication between the firewall and the directory. Elgamal's analogous invention teaches that because the socket layer is widely used in networks, integration of SSL into machines that are connected to the network and receive requests would be facilitated. See column 1, lines 60-63. It would have been obvious to one having ordinary skill in the art at the time the invention was made to take Reid's firewall and use the SSL method taught by Elgamal to communicate between the firewall and the directory. One having ordinary skill in the art at the time the invention was made would have been motivated to do so because Elgamal establishes a need for SSL as a security mechanism between various applications to transfer various data between one another. See column 1, lines 44-54. Furthermore, the nature of the information being transferred between the firewall and the directory in the applicant's invention is private and sensitive.

In reference to claims 7 and 14, which further limit claims 1 and 8 respectively by specifying multiple directories being queried, Reid substantially teaches a system analogous to that in claims 1 and 8 as mentioned above, and also teaches the use of multiple directories as described in column 7, lines 58-61, when he refers to distributed directories and a master directory.

In reference to claims 15 and 16 which further limit claim 8 by specifying that the request comes from an internal user and an external user respectively, Reid substantially teaches a system analogous to that in claim 8 as mentioned above, and

also states that his system handles both internal and external requests. Refer to column 7, lines 39-44, when Reid says that the security policy is defined whether the user is internal or external to the network.

4.3) Claims 3 and 10 are still rejected under 35 U.S.C. 103(a) as being unpatentable over Reid as applied to claims 1 and 8 above, and further in view of US patent number 5,898,830 to Wesinger. Reid discloses a system that encompasses all the limitations of claims 1 and 8, but fails to teach the use of a GUI interface to specify the authorization filter. Wesinger, in an analogous art, teaches the use of a graphical user point and click web interface for configuring the firewall and specifying configuration parameters. It would have been obvious to one having ordinary skill in the art at the time the invention was made to modify the firewall of Reid to include a GUI interface by which the authorization filter could be specified. One having ordinary skill in the art at the time the invention was made would have been motivated to do so because graphical user interfaces have become highly popular and favored for their ease of use, and because they are cheap and easy to develop.

Claims 11 and 12 are still rejected under 35 U.S.C. 103(a) as being unpatentable over Reid in view of Wesinger as applied to claims 3 and 10 above, and further in view of Reid as applied to claims 4 and 5 above.

Conclusion

5.) **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ali M. Mashaal whose telephone number is 703-305-7854. The examiner can normally be reached on 8:30-5:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Ayaz Sheikh can be reached on 703-305-9648. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

AM

AYAZ SHEIKH
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:	)	
Thomas D. Ashoff et al.	)	Examiner: Ali M. Mashaal
Serial No.: 09/495,157	)	Group Art Unit: 2136
Filed: January 31, 2000	)	Docket: 105.201US1
For: SYSTEM, METHOD AND COMPUTER PROGRAM PRODUCT FOR AUTHENTICATING USERS USING A LIGHTWEIGHT DIRECTORY ACCESS PROTOCOL (LDAP) DIRECTORY SERVER	)	

APPELLANTS' BRIEF ON APPEAL UNDER 37 C.F.R. 41.37(c)

Mail Stop Appeal Brief- Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

This Brief is presented in support of the Notice of Appeal mailed July 16, 2004, from the final rejection of claims 1-17 of the above-identified application, as set forth in the Final Office Action mailed February 18, 2004. A copy of the claims being appealed is enclosed as Appendix I.

The Commissioner of Patents and Trademarks is hereby authorized to charge Deposit Account No. 19-0743 in the amount of 250.00 which represents the requisite fee set forth in 37 C.F.R. § 41.2(b)(2).

Appellants respectfully request consideration and reversal of the Examiner's rejections of pending claims 1-17.

03/01/2005 AWONDAF1 00000053 190743 09495157
FC:2402 250.00 DA

APPELANTS' BRIEF ON APPEAL UNDER 37 C.F.R. 41.37(c)

TABLE OF CONTENTS

1. REAL PARTY IN INTEREST

The Real Party in Interest of the above-captioned patent application is Secure Computing Corporation, the assignee of the application.

2. RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences for the above-referenced patent application.

3. STATUS OF CLAIMS

The present application was filed on January 31, 2000 with claims 1-17. A non-final Office Action was mailed September 10, 2003. A Final Office Action was mailed February 18, 2004. Claims 1-17 (Appendix I, Claims) stand rejected under 35 U.S.C. §103(a), remain pending, and are the subject of the present appeal.

4. STATUS OF AMENDMENTS

Appellants have not filed an amendment subsequent to the mailing of the Final Office Action on February 18, 2004.

5. SUMMARY OF CLAIMED SUBJECT MATTER

According to one embodiment, the present invention relates to a system (such as reference numeral 300, Figure 3) for authorizing client (such as reference numeral 102a, Figure 3) access to a network resource (such as reference numeral 118, Figure 3). The system includes a server (such as reference numeral 106, Figure 3; page 7, lines 11-17) that has at least one directory (such as reference numeral 204, Figure 3; page 7, lines 11-17) that can be accessed using a network protocol. The directory is configured to store information concerning an entity's organization (see Figure 4 generally, and page 7, lines 11-17). The system also includes a firewall (such as reference numeral 110, Figure 3; page 9, lines 1-3) that is configured to intercept network resource requests from a plurality of client users. The firewall is operative to authorize a network resource request based upon a comparison (page 9, lines 10-19) of the contents of at least part of one or more entries in the at least one directory to an authorization filter (see page 11, lines 7-15 for discussion relating to authorization filters). The authorization filter is generated based on a directory schema that is predefined by the entity (page 12, line 14-21).

According to another embodiment, the present invention relates to an authentication method (the method is represented by reference numerals 302, 304, 306, 308, and 310 in Figure 3) executed by a firewall (such as reference numeral 110 in Figure 3). The method includes receiving a network authorization request (such as reference numeral 304, Figure 3; page 8, line 20-page 9, line 2) from a client user (such as reference numeral 102a in Figure 3; page 8, line 20-page 9, line 2). The method also includes querying (reference numeral 306, Figure 3; page 9,

lines 10-19), using a network protocol, at least one directory (reference numeral 204, Figure 3; page 9, lines 10-19) that is configured to store information concerning an entity's organization (see Figure 4, generally; page 7, lines 11-17). The query is based upon an authorization filter (see page 11, lines 7-15 for discussion relating to authorization filters) that is generated based on a directory schema that is predefined by said entity (page 12, line 14-21). The method further includes determining, based on the results of the query (reference numeral 308, Figure 3; page 9, lines 10-19), whether the contents of at least part of one or more entries in said at least one directory satisfy the authorization filter (page 9, lines 10-19). Finally, the method includes permitting the network resource request through the firewall if the authorization filter is satisfied (page 9, lines 17-19).

According to another embodiment, the present invention relates to a program product for enabling a processor in a computer system to implement an authentication process (the process is represented by reference numerals 302, 304, 306, 308, and 310 in Figure 3). The program product includes a computer usable medium having computer readable program code embodied in the medium for causing a program to execute on the computer system. The program code includes a first computer readable program code for enabling the computer system (such as reference numeral 110, Figure 3; page 8, line 20-page 9, line 2) to receive a network request from a client user (such as reference numeral 102a, Figure 3; page 8, line 20-page 9, line 2). The program code also includes a second computer readable program code for enabling the computer system to query (reference numeral 306, Figure 3; page 9, lines 10-19), using a network protocol, at least one directory (reference numeral 204, Figure 3; page 9, lines 10-19) that is configured to

store information concerning an entity's organization (see Figure 4, generally; page 7, lines 11-17). The query is based upon an authorization filter (see page 11, lines 7-15 for discussion relating to authorization filters) that is generated based on a directory schema that is predefined by said entity (page 12, line 14-21). The program code also includes a third computer readable program code for enabling the computer system to determine, based on the results of the query (reference numeral 308, Figure 3; page 9, lines 10-19), whether the contents of at least part of one or more entries in the at least one directory satisfy the authorization filter (page 9, lines 10-19). Finally, the program code also includes fourth computer readable program code for enabling the computer system to permit the network resource request through the firewall if the authorization filter is satisfied (page 9, lines 17-19).

This summary does not provide an exhaustive or exclusive view of the present subject matter, and Appellant refers to the appended claims and their legal equivalents for a complete statement of the invention.

6. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Whether independent claims 1-17 have been erroneously rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 6,131,120 to *Reid* in view of the *Microsoft Computer Dictionary* 1997, in further view of *Check Point Account Management Client*, Version 1.0, September 1998, and in view of other art relevant to the dependent claims (not at issue herein).

7. ARGUMENT

A. The Law Applicable Under 35 U.S.C. §103

MPEP §2142 states the basic applicable law governing obviousness of claimed subject matter:

> To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, and not based on applicant's disclosure. *In re Vaeck*, 947 F.2d 488, 20 USPQ2d 1438 (Fed. Cir. 1991).

B. Introduction

Claims 1-17 each include a limitation requiring a comparison between data from a directory and an authorization filter. None of the prior art cited in the Office actions to date teach or suggest such a comparison. For this reason, claims 1-17 should have been allowed.

C. Appellants' Invention

Appellants describe, and claim in claims 1-17, a firewall that communicates with a directory hosted on a server to determine if a network resource access request should be authorized or denied. Figure 1 depicts an example of such a firewall. The example presented in Figure 1, and discussed throughout, is intended to generally familiarize the reader with Appellants' invention, and is not intended to be an exhaustive description.

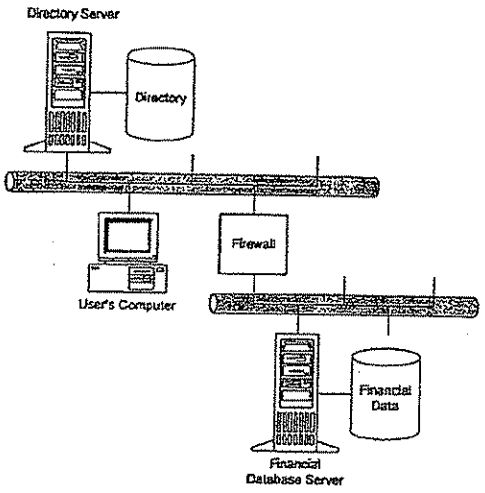


Figure I

As can be seen from Figure 1, a firewall is interposed between user's computer and a financial database server, an example of a "network resource." Thus, any attempts to access the financial database server must pass through the firewall. The firewall intercepts the computer's first attempt to access the financial database server. See Application, page 9, lines 1-3.

In the wake of having intercepted the request, the firewall performs an operation to identify the user logged on to the computer from which the request emanated. See Application, page 9, lines 3-13. Based upon the identity of the user, the firewall queries a directory stored on a server to learn of attributes describing the user. Id.

A directory is similar to a database, in that it stores data that may be retrieved via query, but a directory is tailored to be read from more than it is written to. In the case of this invention, the directory stored on the server contains information concerning the organization of the entity

employing the computer user. See Application, page 7, lines 11-17. For example, the directory may be organized as shown in Figure 2.

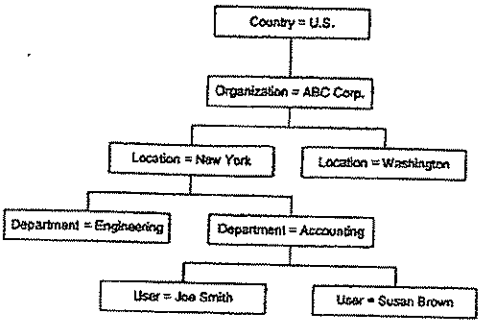


Figure 2

Thus, by virtue of querying such a directory with a given user's name, the firewall may obtain the department employing the user, the location in which the user is employed, the name of the organization employing the user, and the country in which the user is employed. Each of these pieces of information is an attribute describing the user.

After having obtained the user's attributes, the attributes are compared to an authorization filter. At its simplest, an authorization filter is an attribute and value pair. In the case of this example, the filter may be "Department = Accounting" (i.e., the user must work for the accounting department to be able access the financial database server). The firewall authorizes or denies the request on the basis of this comparison.

The act of comparing the authorization filter to data obtained from the directory is an element of every independent claim. For example, claim 1 requires "a comparison of the

contents of at least part of one or more entries in said at least one directory to an authorization filter." Similarly, claims 8 and 17 require determining "whether the contents of at least part of one or more entries in said at least one directory satisfy said authorization filter." Thus, the independent claims require the general structure depicted in Figure 3, below

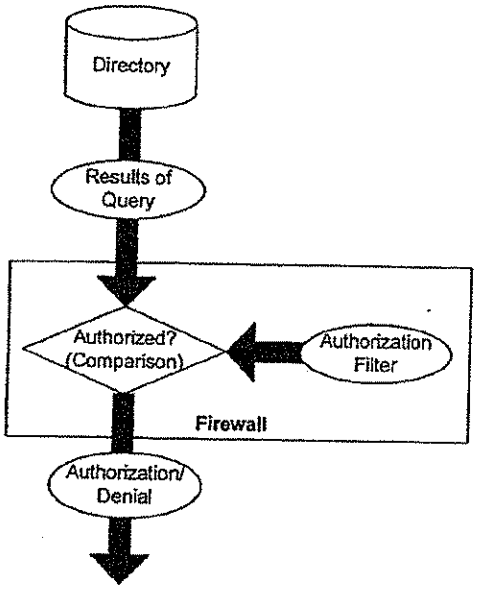


Figure 3

As shown in Figure 3, the claims require the authorization process to include a comparison of data returned from a query of the directory to an authorization filter.

D. The Prior Art

1. Summary of the Cited Prior Art

The prior art cited against the independent claims includes: (1) U.S. Patent No. 6,131,120 to *Reid*; (2) *Microsoft Computer Dictionary* 1997; and (3) *Check Point Account Management Client*, Version 1.0, September 1998. Of these, only the teachings of *Reid* are at issue. Briefly, *Microsoft Computer Dictionary* is cited to for its definition of the term "firewall," in order to support the proposition that the routers in *Reid* may be thought of as firewalls, because of the functionality they provide (Appellants do not dispute this). *Check Point Account Management Client* is a software manual, and is cited to support the proposition that it would have been obvious to modify the directory described in *Reid* to store information concerning an entity's organization (for the purposes of this appeal only, Appellants do not dispute this).

2. U.S. Patent No. 6,131,120 to *Reid*

Reid teaches a system whereby each router or gateway (collectively referred to herein as a router) stores a router access list. A router access list is a list of names and addresses of users and the destinations they are permitted to access. See Final Office Action, page 2 ("Each RAL [router access list] . . . includes the names and addresses of users and the destinations they are allowed to reach."). According to the scheme taught in *Reid*, a router intercepts a network resource access request, and extracts the requesting computer's address and requested destination address. If the router access list indicates that the requesting computer is permitted to access the requested address, the access is authorized, otherwise it is denied. See Final Office Action, pages 2-3.

The router access list taught by *Reid* is created automatically by a software application running on a server that stores a directory. See *Reid*, col. 8, lines 23-28. The directory contains, amongst other entries, the name and access privileges assigned to each user. See *Reid*, col. 8, lines 6-11 and 23-27. For each router in an organization's system, the application creates an appropriate router access list and sends the list to the router. See *Reid*, col. 8, lines 21-34.

The general structure of the authorization process described by *Reid* is depicted in Figure 4, below.

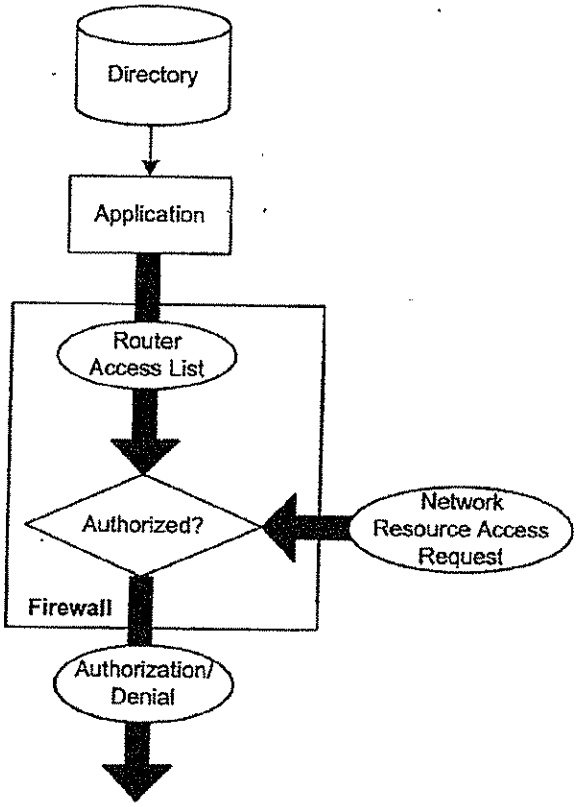


Figure 4

E. Brief Summary of Prosecution

The above-captioned application was filed on January 31, 2000. A first Office action was mailed on September 10, 2003 (claims 1-17 were rejected under 35 U.S.C. §103(a) as being obvious). Appellants filed a response on January 16, 2004. A final Office action was mailed on February 18, 2004 (claims 1-17 remained rejected under 35 U.S.C. §103(a), although the premise of the rejection changed). Finally, Appellants filed a notice of the present appeal on July 16, 2004.

F. Appellants' Rebuttal of the Rejections of the Independent Claims

1. The Initial Office Action

In a first Office action mailed September 10, 2003, the independent claims were rejected as being obvious in light of *Reid* and other prior art, the teachings of which are not presently at issue.

As stated above, each of the independent claims requires the act of comparing an authorization filter to data obtained from the directory. On its face, *Reid* wholly lacks any teaching or suggestion of such an act. To formulate his initial rejection, the Examiner seized upon an incorrect interpretation of *Reid*. Specifically, the Examiner assumed the existence of two separate unarticulated steps in Reid: (1) that the firewall transmitted access criteria, which the Examiner likened to an authorization filter, to the directory; and (2) that a comparison between the access criteria and the contents of the directory was performed as a part of the process of producing the router access lists. According to the Examiner,

> . . . in order for the directory to generate the appropriate [router] access list, each router/gateway must have transmitted its access criteria to the directory. The examiner further asserts that this criteria is an authorization filter and that in order for the directory to send back a correct access list, some comparison must have been made with directory entries and the router/gateway criteria (authorization filter).
>
> Initial Office Action, page 5.

Thus, according to the Examiner, the authorization process of *Reid* includes the extra—albeit completely unmentioned—steps, shown in dashed lines, depicted in Figure 5, below.

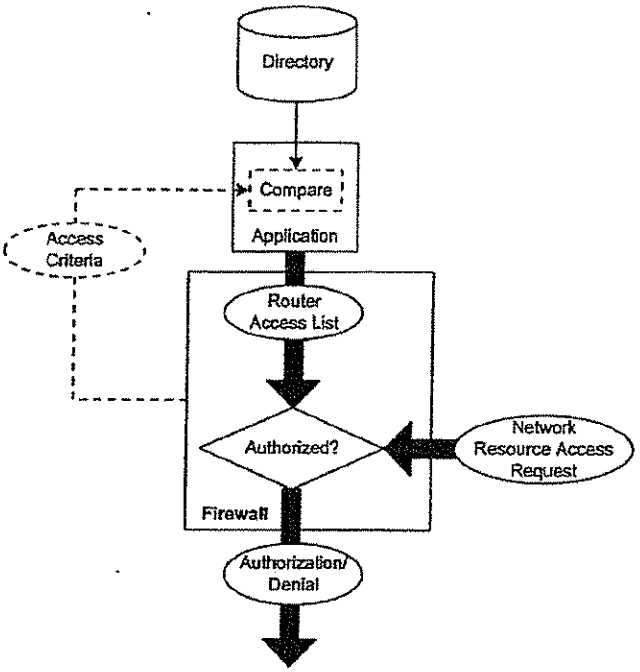


Figure 5

In a response filed January 16, 2004, Appellants argued that neither the "access criteria," nor the comparison step are referred to, or even hinted at, by the text of *Reid*. Further, Appellants

went on to argue that the text of *Reid* plainly contradicts the very notion of their existence. Appellants argued that the router access lists of *Reid* are generated on the sole basis of the contents of the directory. Put simply, the application reads the directory and creates the router access lists. The process involves no transmission of access criteria, nor does it involve a comparison between access criteria and the contents of the directory.

In support of their position, Appellants cited column 6, lines 23-25 of *Reid* (emphasis added), which states that entries in the directory control the entire network:

> Because the directory knows the location and IP address of each user, and the location and IP address of each router/gateway, a directory application can periodically populate the RAL [router access list] in each router/gateway on the network using LDAP. Entries in the directory thereby control the entire network and the network router/gateway configuration management is automated.

The Examiner seems to have accepted this argument, because, in authoring his final rejection, he has altered his interpretation of *Reid*, in order to find a new way in which one might understand it to include a comparison between an authorization filter and data obtained from the directory.

2. The Final Office Action

In authoring his final rejection, the Examiner has reinterpreted *Reid* to find the missing element in a routine act performed by virtually every router that has been configured to act as a firewall: the act of comparing a network access request to a router access list. According to the Examiner, the routers of *Reid*

> compare[] the requesting device's address and requested destination to that information in the router/gateway which was provided by the directory server [i.e., the router access list], in order to determine whether the requesting device should be allowed/denied access. Therefore, the router/gateway clearly contains an authorization filter by which it can make a comparison of the content of at least

> part of one or more entries in the directory to determine which traffic may be allowed to pass through to a given destination.
>
> Final Office Action, pages 2-3.

The Examiner's latest interpretation of *Reid* still falls short of requirements posed by each of Appellants' claims. The independent claims require a comparison between an authorization filter and at least a portion of an entry in the directory. For example claims 1-7 require "a comparison of the contents of at least part of one or more entries in said at least one directory to an authorization filter." (Claims 8 through 17 include similar language.) The Examiner's interpretation does not even purport to find such a comparison in *Reid*. Instead, the Examiner's interpretation of *Reid* yields a comparison between a router access list (which the Examiner likens to an authorization filter) and address data extracted from the network access request. The difference between the claimed invention and the Examiner's latest interpretation of Reid is depicted in Figure 6.

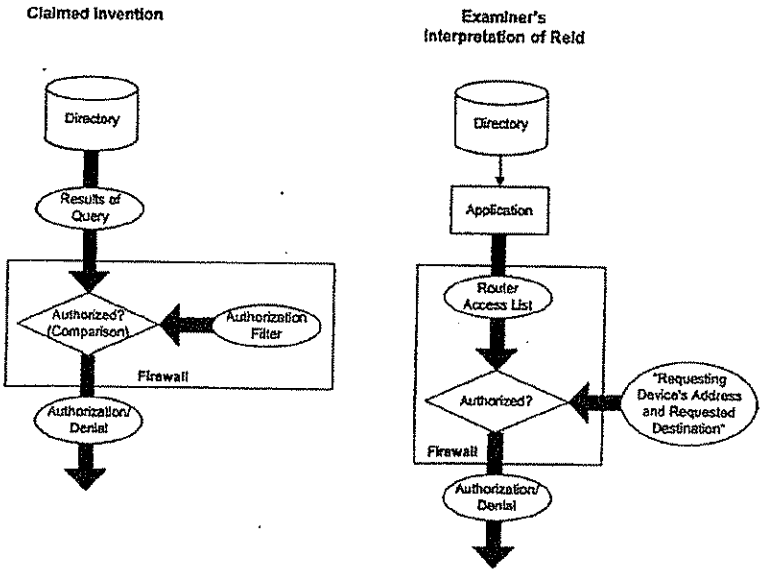


Figure 6

As can be seen from Figure 6, the claimed invention includes a comparison between an authorization filter and at least a portion of an entry from the directory (i.e. the "results of query" shown in Figure 6). Turning to *Reid*, on the other hand, even if one accepts the Examiner's position that a router access list is an authorization filter, the comparison does not occur between the proper subjects. Therein, at stated in the Examiner's Response to Arguments, the comparison occurs between the router access list (asserted to be an authorization filter) and "the requesting device's address and requested destination." See Final Office Action, page 2. The requesting device's address and requested destination are units of information extracted from the network resource access request—not from the directory. Therefore, at best, the comparison in *Reid* occurs between an authorization filter (router access list) and information extracted from the

network resource access request[1], not the query results as described by Appellants and stated in claims 1-17. Plainly, such a comparison does not rise to textual requirements presented in claims 1-17.

[1] For the present purposes, Appellants take no position regarding whether a router access list may be properly characterized as an authorization filter. To the extent Appellants have appeared to acquiesce in this portrayal, Appellants have done so in order to present the Examiner's interpretation of *Reid*—not to endorse the Examiner's interpretation.

G. Conclusion

The prior art cited against claims 1-17 fails to teach or suggest a comparison between an authorization filter and data obtained from a directory. This act is required by each of the rejected claims. The record fails to present any motivation for one to modify *Reid* to include such a step. Indeed, it is not clear how *Reid* could so be modified, without altering the theory of operation of the firewall function presented therein. For at least this reason, the rejection of claims 1-17 is improper, and the rejection should be withdrawn.

Respectfully submitted,

THOMAS D. ASHOFF ET AL.

By their Representatives,

SCHWEGMAN, LUNDBERG, WOESSNER & KLUTH, P.A.
P.O. Box 2938
Minneapolis, MN 55402

Date 22 Feb. 2004 By

Nicholas P. Johns
Reg. No. 48,995

CERTIFICATE UNDER 37 CFR 1.8: The undersigned hereby certifies that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail, in an envelope addressed to: Mail Stop Appeal Brief, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on this 22 day of February, 2005.

Peter Rebuffoni
Name

Signature

APPENDIX I

THE CLAIMS ON APPEAL

1. (Previously Presented) A system for authorizing client access to a network resource, comprising:

a server having at least one directory that can be accessed using a network protocol, said at least one directory being configured to store information concerning an entity's organization; and

a firewall that is configured to intercept network resource requests from a plurality of client users, said firewall being operative to authorize a network resource request based upon a comparison of the contents of at least part of one or more entries in said at least one directory to an authorization filter, wherein said authorization filter is generated based on a directory schema that is predefined by said entity.

2. (Original) The system of claim 1, wherein said at least one directory is a lightweight directory access protocol directory.

3. (Original) The system of claim 1, wherein said authorization filter is specified using a graphical user interface.

4. (Original) The system of claim 1, wherein said authorization filter implements a per-user authentication scheme.

5. (Original) The system of claim 1, wherein said authorization filter implements a per-service authentication scheme.

6. (Original) The system of claim 1, wherein said firewall and said directory communicate using secure socket layer communication.

7. (Original) The system of claim 1, wherein said firewall is configured to query multiple directories.

8. (Original) An authentication method at a firewall, comprising the steps of:

(a) receiving a network resource request from a client user;

(b) querying, using a network protocol, at least one directory that is configured to store information concerning an entity's organization, wherein said query is based upon an authorization filter that is generated based on a directory schema that is predefined by said entity;

(c) determining, based on the results of said query, whether the contents of at least part of one or more entries in said at least one directory satisfy said authorization filter; and

(d) permitting said network resource request through said firewall if said authorization filter is satisfied.

9. (Original) The method of claim 8, wherein step (b) comprises the step of querying said at least one directory using a lightweight directory access protocol.

10. (Original) The method of claim 8, further comprising the step of specifying an authorization filter using a graphical user interface.

11. (Original) The method of claim 10, wherein said specifying step comprises the step of specifying an authorization filter that implements a per-user authentication scheme.

12. (Original) The method of claim 10, wherein said specifying step comprises the step of specifying an authorization filter that implements a per-service authentication scheme.

13. (Original) The method of claim 8, wherein step (b) comprises the step of querying said directory using secure socket layer communication.

14. (Original) The method of claim 8, wherein step (b) comprises the step of querying multiple directories.

15. (Original) The method of claim 8, wherein step (a) comprises the step of receiving a network resource request from a client user at an internal network.

16. (Original) The method of claim 8, wherein step (a) comprises the step of receiving a network resource request from a client user at an external network.

17. (Original) A computer program product for enabling a processor in a computer system to implement an authentication process, said computer program product comprising:

a computer usable medium having computer readable program code embodied in said medium for causing a program to execute on the computer system, said computer readable program code comprising:

first computer readable program code for enabling the computer system to receive a network resource request from a client user;

second computer readable program code for enabling the computer system to query, using a network protocol, at least one directory that is configured to store information concerning an entity's organization, wherein said query is based upon an authorization filter that is generated based on a directory schema that is predefined by said entity;

third computer readable program code for enabling the computer system to determine, based on the results of said query, whether the contents of at least part of one or more entries in said at least one directory satisfy said authorization filter; and

fourth computer readable program code for enabling the computer system to permit said network resource request through said firewall if said authorization filter is satisfied.

IEE/IEE Electronic Library online

CITATION AND ABSTRACT

HOME | SEARCH | ADVANCED SEARCH | WHAT'S NEW | HELP | LOGOUT
SEARCH RESULTS | VIEW FULL PAGE | PREVIOUS RECORD |

Secure code distribution

- *Zhang, X.N.*
955 Escalon Avenue, Sunnyvale, CA, USA

This Paper Appears in :
Computer

on Pages: 76 - 79

June 1997 Vol. 30 Issue: 6 ISSN: 0018-9162

References Cited: 11
CODEN: CPTRB4
Accession Number: 5604190

Abstract:

The Java Virtual Machine does not offer a way for code obtained from trusted sources to be granted extra rights. The article describes two approaches to authentification for code distribution: one extends the JVM to include a digital signature in applets; the other uses MIME encapsulation to take advantage of available security infrastructures. The signed-applet approach gives a programmer more flexibility because it addresses the security issues at a more fundamental level. However, signed-applet security mechanisms may vary for different code distribution schemes, making integration difficult. The MIME-based approach provides a unified security interface. It is more efficient in the sense that all classes can be encapsulated in one multipart attachment, and a single signature or verification operation will cover all classes. The approaches can also be combined and tailored to satisfy various requirements. Ultimately, operating systems must support the concept of a secure compartment so that separate resource management policies can be implemented for the secure compartment and the rest of the system.

Subject Terms:

JA1228

Secure Code Distribution

With code proliferating on the Web, [illegible] emerging as major issues [illegible] intentional or unintentional damage. [illegible]

X. Nick Zhang
EIT

While the Web has made the distribution of executable code far more convenient, it has also given rise to various problems. The most serious is security. Not so long ago you simply placed your trust in a few software companies, whose developers took care that no harmful materials could be inserted into their products. But all this has changed.

First and foremost, the relative ease of programming on the Web makes code prolific. With Java applets, JavaScript, ActiveX, and other Web-oriented technologies, almost anyone can put code in a Web page for someone else to download. Moreover, a growing number of software vendors are marketing their products online to slash distribution costs. In addition to Web-based code distribution in the pull model, the idea of active messaging is being tested with the push model through e-mail-based code distribution.[1] This proliferation of code makes security checking extremely difficult for the receiver. Besides, no strict boundary separates passive data from active code. Even a piece of a Microsoft Word macro can cause undesirable effects.

A subtle difference distinguishes code distribution from distributed programming. The latter focuses on method invocation rather than passing autonomous objects, and its security problems are different from those of code distribution.[2] For example, access control is considered mainly on the client side in code distribution but on the server side in distributed programming.

In describing several different approaches to secure code distribution, this article follows Java's terminology conventions. Thus the mobile code is always called an applet and the security gatekeeper is called the SecurityManager. This is because the Java applet is currently the most popular code distribution scheme, and its security model has been subjected to various security screenings and is thus the most sophisticated model so far.

CODE DISTRIBUTION MECHANISMS

In Safe-Tcl,[1] one of the earliest code distribution efforts, the enabled or active messages are classified into four categories based on phases: submission, delivery, receipt, and activation. These phases relate to an applet's life cycle, as illustrated in Figure 1.

Submission time is when the message is submitted to the user agent. Delivery and receipt phases must contend with firewalls on both ends and may be significantly affected by firewalls or proxies. For example, a discretionary proxy may not let applets pass through. Activation occurs after authorization by the SecurityManager, which constitutes the last line of defense for the secure compartment housing some of the system resources.

Other code distribution mechanisms may have minor variations in their implementation. In directly interpreted scripting languages, bytecode may never be generated because compilation and interpretation both occur on the recipient's machine. In Safe-Tcl, the trusted interpreter is the security manager, which never directly evaluates untrusted code. Instead, a procedure defined in the trusted interpreter filters out untrusted code and exports the result to the untrusted interpreter.

SECURITY ISSUES

Two fundamentally different security issues affect code distribution:

- *Authentication.* A recipient cannot simply trust the code source. Moreover, the message containing the applet may be compromised by someone else between delivery and receipt. Conventional (or passive) messaging shares this problem.
- *Protection/safety.* Applets may contain harmful contents—for example, instructions to access local resources not intended for them. Activation-time mechanisms, which have their own security models, should be used to combat this problem.

The Java Virtual Machine is designed to control potentially dangerous access to system resources during activation time so that untrusted code can be safely executed locally. However, it does not offer a way for code obtained from trusted sources to be granted extra rights to perform operations that are forbidden to less trusted sources. For example, current versions of the Netscape browser do not let an applet access the local file system or connect a host

0018-9162/97/$10.00 © 1997 IEEE

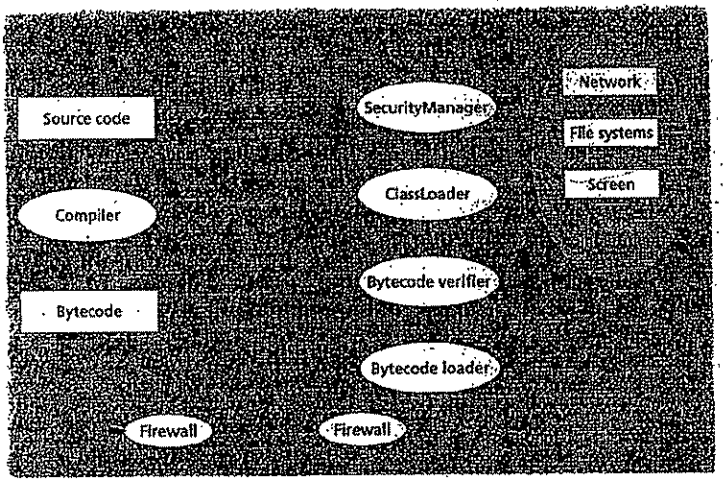


Figure 1. Security model of an applet showing four phases related to the applet's life cycle.

other than the one it comes from. This model of security limits the practical functionality of mobile code. In addition, code distribution doesn't have to take the form of server-to-client download; it could also operate in reverse, uploading *servlets* from client to server. This makes the issue of trust particularly important.

With authentication and (optionally) privacy, a trust relationship can be established in a way that gives trusted code more privileges. In a public-key-based encryption scheme, authentication often takes the form of a digital signature. The message is run through a one-way hash function, and the result is encrypted with the sender's private key. The encrypted hash—the message's signature—is sent along with the message. The recipient retrieves the sender's public key, which may be stored in a global directory infrastructure and signed by a certificate authority, and uses it to decrypt the hash value and see if it matches the locally generated hash result. Bruce Schneier's book is an excellent source on encryption technologies.[3]

The signature is a deterrent to damage because the sender and/or author of a malicious applet can be traced. When notified of signature verification, the SecurityManager can determine what access privileges to grant the incoming applet. Trusted applets enable more interesting applications.

Authentication

Below I describe two approaches to authentication for code distribution. One extends the Java Virtual Machine to include a digital signature in applets; the other uses MIME (Multipurpose Internet Mail Extension) encapsulation to take advantage of available security infrastructures. These approaches can be used separately or together to implement desired security policies.

Signed applet. The Java class file format has an extensible attribute syntax.[4] New attributes can be added to a class file without breaking existing class parsers. The idea of a signed applet is simply to extend the class file with several new attributes into which a digital signature can be embedded. These attributes must be designed to meet several requirements:

- For efficiency, the number of public key operations must be kept to a minimum because they are very time-consuming.
- A signed class must be able to reference other signed class files.
- It must be possible for classes to be signed in any order.

In the extended class file, two strings are added to the end of the constant pool. These strings are used as names for the two new attributes: HashBlock and SignatureBlock. If an applet consists of many individually signed utility classes, loading these classes at the receiving end may result in a significant slowdown caused by verification of referenced utility classes. The efficiency requirement is satisfied by including the hash results of referenced classes as part of a signed class and storing them in the HashBlock attribute. Since these utility classes can be used separately, a signed class may reference other signed classes. The signatures are placed in the SignatureBlock attribute. However, this procedure complicates the hash calculation for referenced classes. Since referenced classes can be signed, and this signing may occur after the

A new MIME type has been proposed to facilitate authentication in both the push mode for e-mail and the pull mode for the Web.

indirect hash is taken, changes caused by the signature process may show up when the hash of referenced classes is calculated. Such changes must be undone.

The HashBlock contains direct hashes, that is, hashes obtained by running the hash function on the original class files instead of the extended ones. If a HashBlock already exists in a previously signed class file, the signing process must not remove or alter it in any way unless it also removes any preexisting SignatureBlock. The process should verify all the items in the HashBlock. It should also refuse to sign the class if the verification of one of these items fails or if no hash exists for a class that requires one, as determined by the signing process.

The SignatureBlock contains all the signatures. If a SignatureBlock already exists in a previously signed class, the new signature should be added to the existing block. The hash used for the digital signature is much simpler than that used for hash blocks; it covers the entire class file except the SignatureBlock. As a matter of fact, these fields form a contiguous block of data; thus the hash function need only be updated twice.

Although a class can be trusted not to do anything harmful when called by the top-level application or another trusted class, it might not check all its arguments well enough to prevent harm when called by an untrusted program. A special flag placed in the signature can be set to signify the signer's assertion that the class checks all its arguments. Any security checks that involve walking up a call chain can terminate at this point. If this flag is not set, the SecurityManager must continue up the chain until it reaches the top level or hits a class that does have the flag set, in which case the check succeeds.

Secure code distribution in MIME. Ali Bahreman and colleagues[5] have proposed a new MIME type, application/applet, to encapsulate information needed to transport an applet. This approach is promising in both the push mode for e-mail and the pull mode for the Web. Several parameters must be in place in the applet object's MIME header. For instance, a name parameter must appear in the content-type field to identify the real class names for the applet and related classes. The site parameter specifies where the applet comes from. This information must be conveyed to the SecurityManager, which will process the applet and use the local security policy to determine whether to grant additional privileges. Since not all applets may require network connection, the site parameter is optional.

MIME provides a container content type called multipart.[6] A multipart message is essentially a folder that can hold any number of attachments. Each attachment is a MIME object that may be recursively typed as multipart. In Safe-Tcl,[3] the MIME object, typed as multipart/enabled-mail, contains two attachments, one for the code and the other for parameters. However, this arrangement is inadequate for general-purpose code distribution. A Java applet may refer to other classes that must all be encapsulated at the same time. The content-type multipart/related was originally proposed to encapsulate multiple interreferenced HTML documents.[7] We can extend it to become a general-purpose container for the applet and the applet's input data, as well as for other referenced classes. The detailed technical specification for MIME encapsulation exists as an Internet draft.[5]

Since an applet can be considered a data object within an HTML document, a tight coupling between the work on MIME encapsulation of aggregate HTML documents[8] and MIME encapsulation for transporting applets may be possible.

Almost all secure messaging protocols, such as S-HTTP, MIME Object Security Services (MOSS), S/MIME, and MIME/PGP, rely on the flexibility of multipart. Together with messages that must be secured, protocol information—such as digital signatures and encrypted session keys—can be placed into another multipart-typed envelope. The MOSS specification[9] describes the framework within which security services may be applied to MIME body parts. You can use multipart/signed and multipart/encrypted objects to secure other MIME objects. For the applet to be signed, the applet class files are first encapsulated in a multipart/related object, which in turn is signed by MOSS and put into a multipart/signed object. The applet's recipient should verify the MOSS signature before invoking the applet. Encrypting the compound applet object with MOSS will reduce the concern over someone eavesdropping.

In theory, applets and related classes can be processed individually with certain secure MIME mechanisms before being put into a container object, but there are two reasons for not doing so: First, it will complicate the semantics of multipart/related; second, it is more efficient to securely enhance an applet with its related classes all at the same time by putting them into a single security envelope.

Protection/safety

Authentication doesn't entirely solve the safety problem because there is no consensus on the legal, social, and technological meanings of trust. Suppose you download an applet from a trusted source and it "accidentally" removes an important file from your machine. What then? Trust didn't make you safe.

Java's strong type-checking makes damaging code less likely to be written in the first place, at least unintentionally. Scripting languages such as Safe-Tcl and Safe-Perl may be ideal alternatives to Java for code distribution because of their wide popularity. Although they do not perform type-checking in the conventional sense, the fact that they have inherently one

type—string—generally makes type-checking unnecessary. They have an activation-time security model, sometimes called a sandbox model, similar to Java's. Their security manager confines all possible activities inside the secure compartment. Of course, this limits the applet's functionality.

The Princeton Group proposed[10] a helpful safety mechanism: Each time a local file access is needed, SecurityManager queries the user for permission. However, in many applications such intensive user involvement doesn't seem viable. CORBA Mobile Agent Facility,[2] for example, emphasizes interactions among programs instead of humans.

The sandbox model's fundamental problem is that the secure compartment resides inside the system, but if the user's resources are to be protected, the secure compartment must be isolated from the workspace. For example, a file system dedicated to untrusted code could be established in a way that separates its activity from the file system in the workspace. With SecurityManager, the user could conceptually build a personal firewall between the secure compartment and the rest of the system. The secure compartment would be like a guest room in a house. In the worst case, even if some damage occurred in the "guest room," it would not directly affect how the workspace functions.

A general-purpose secure code distribution scheme must be flexible to accommodate as many mechanisms as possible. It must operate efficiently, and not significantly slower than nonsecure schemes. Ideally, it would use existing security infrastructures, because building a new public-key infrastructure could be extremely time-consuming and would shift the focus from code distribution.

The signed-applet approach gives a programmer more flexibility because it addresses the security issues at a more fundamental level. However, signed-applet security mechanisms may vary for different code distribution schemes, making integration with other secure messaging protocols difficult.

In contrast, the MIME-based approach provides a unified security interface. It is more efficient in the sense that all classes can be encapsulated in one multipart attachment, and a single signature or verification operation will cover all classes. Microsoft's Authenticode[11] and JavaSoft's JAR already offer these features, although they use proprietary encapsulation formats. In addition, MIME-based active messaging can take advantage of an existing infrastructure and several existing MIME-based security mechanisms, such as S/MIME and MOSS.

The signed-applet and MIME-based approaches can be combined and tailored to satisfy various requirements. For example, an author might sign an applet and put it into a MIME envelope encrypted and/or signed with MOSS by another party before it is sent to the recipient.

Ultimately, operating systems must support the concept of a secure compartment so that separate resource management policies can be implemented for the secure compartment and the rest of the system. ✦

Authentication alone won't make you safe. A downloaded applet from a trusted source could still do some damage.

Acknowledgments

I thank Ali Bahreman, Jim Galvin, Simon Spero, and Rajkumar Narayanaswamy for productive discussions.

References

1. N. Borenstein, "EMail with a Mind of Its Own: The Safe-Tcl Language for Enabled Mail," *Proc. IFIP WG 6.5 Conf.*, Elsevier (North Holland), 1994.
2. IBM, *Mobile Agent Facility Specification*, OMG TC Document ftp://ftp.omg.org/pub/docs/cf/96-12-01.pdf, Dec. 1996.
3. B. Schneier, *Applied Cryprography*, second ed., John Wiley & Sons, New York, 1996.
4. T. Lindholm and F. Yellin, *Java Virtual Machine Specification*, Addison-Wesley, Reading, Mass., 1996.
5. A. Bahreman et al., "MIME Encapsulation of Aggregate Applet Objects," Internet draft, also available from http://ecn.cit.com/mapplet/mapplet.html, June 1996.
6. N. Borenstein and N. Freed, "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies," Internet RFC 1521, http://ds.internic.net/rfc/rfc1521.txt, Sept. 1993.
7. E. Levinson, "The MIME Multipart/Related Content-Type," Internet RFC 2112, http://ds.internic.net/rfc/rfc2112.txt, Mar. 1997.
8. J. Palme and A. Hopmann, "MIME E-Mail Encapsulation of Aggregate Documents, such as HTML (MHTML)," Internet RFC 2110, http://ds.internic.net/rfc/rfc2110.txt, Mar. 1997.
9. J. Galvin et al., "MIME Object Security Services," Internet RFC 1847-1848, http://ds.internic.net/rfc/rfc1848.txt, Oct. 1995.
10. D. Dean, E.W. Felten, and D.S. Wallah, "Java Security: From HotJava to Netscape and Beyond," *Proc. IEEE Symp. Security and Privacy*, IEEE CS Press, Los Alamitos, Calif., 1996.
11. Microsoft, Authenticode Technology White Paper, http://www.microsoft.com/security, Oct. 1996.

X. Nick Zhang is a senior software engineer at EIT. His interests include logic, history, and Peking Opera. Zhang received an MS in computer science from the University of Massachusetts at Amherst and is completing work to earn an MBA from West Virginia University.

Contact Zhang at 955 Escalon Avenue, Sunnyvale, CA 94086; zhang@eit.com.

JA1233 THROUGH JA1281
REDACTED